PCT ORGANISATION MONDIALE DE LA PROPRIETE INTELLECTUELLE
Bureau international

DEMANDE INTERNATIONALE PUBLIEE EN VERTU DU TRAITE DE COOPERATION EN MATIERE DE BREVETS (PCT)

(51) Classification internationale des brevets 6 :		(11) Numéro de publication internationale:	WO 99/35573
G06F 9/46	A1	(43) Date de publication internationale:	15 juillet 1999 (15.07.99)

(21) Numéro de la demande internationale: PCT/FR98/02886

(22) Date de dépôt international: 28 décembre 1998 (28.12.98)

(30) Données relatives à la priorité:
97/16699 30 décembre 1997 (30.12.97) FR

(71) Déposant *(pour tous les Etats désignés sauf US)*: BULL S.A. [FR/FR]; 68, route de Versailles, F–78434 Louveciennes Cedex (FR).

(72) Inventeurs; et
(75) Inventeurs/Déposants *(US seulement)*: BAILLIF, Christian [FR/FR]; 7 bis, avenue du Petit Chambord, F–92340 Bourg la Reine (FR). DIA, Mama, Saidou [FR/FR]; 181, avenue Jean Jaurès, F–92290 Chatenay Malabry (FR).

(74) Mandataire: BERTRANDIAS, Patricia; Bull S.A., 68, route de Versailles, PC58F35, F–78434 Louveciennes Cedex (FR).

(81) Etats désignés: US, brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Publiée
Avec rapport de recherche internationale.
Avant l'expiration du délai prévu pour la modification des revendications, sera republiée si des modifications sont reçues.

(54) Title: METHOD FOR ASSISTING THE ADMINISTRATION OF A DISTRIBUTED APPLICATION BASED ON A BINARY CONFIGURATION FILE IN A COMPUTER SYSTEM

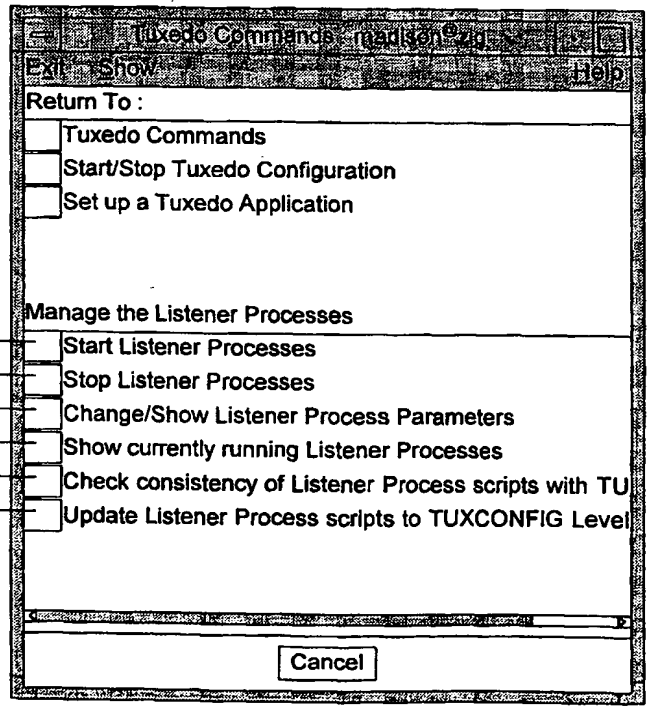
(54) Titre: PROCEDE D'ASSISTANCE A L'ADMINISTRATION D'UNE APPLICATION DISTRIBUEE BASEE SUR UN FICHIER BINAIRE DE CONFIGURATION DANS UN SYSTEME INFORMATIQUE

(57) Abstract

The invention concerns a method for assisting the administration of a distributed application of a transaction processing manager based on a binary configuration file (TUXCONFIG) characterised in that said method consists in: decompiling the master machine active configuration file; recovering data in the master machine decompiled configuration file; verifying the consistency of said application operated on the machine concerned.

(57) Abrégé

La présente invention concerne un procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions basée sur un fichier binaire de configuration (TUXCONFIG) caractérisé en ce que ledit procédé comporte: une étape de décompilation du fichier de configuration actif de la machine maître (Mm), une étape de récupération d'informations dans le fichier de configuration décompilé de la machine maître, une étape de vérification de la consistance de ladite application mise en oeuvre sur ladite machine donnée.



UNIQUEMENT A TITRE D'INFORMATION

Codes utilisés pour identifier les Etats parties au PCT, sur les pages de couverture des brochures publiant des demandes internationales en vertu du PCT.

AL	Albanie	**ES**	Espagne	**LS**	Lesotho	**SI**	Slovénie
AM	Arménie	**FI**	Finlande	**LT**	Lituanie	**SK**	Slovaquie
AT	Autriche	**FR**	France	**LU**	Luxembourg	**SN**	Sénégal
AU	Australie	**GA**	Gabon	**LV**	Lettonie	**SZ**	Swaziland
AZ	Azerbaïdjan	**GB**	Royaume-Uni	**MC**	Monaco	**TD**	Tchad
BA	Bosnie-Herzégovine	**GE**	Géorgie	**MD**	République de Moldova	**TG**	Togo
BB	Barbade	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tadjikistan
BE	Belgique	**GN**	Guinée	**MK**	Ex-République yougoslave de Macédoine	**TM**	Turkménistan
BF	Burkina Faso	**GR**	Grèce			**TR**	Turquie
BG	Bulgarie	**HU**	Hongrie	**ML**	Mali	**TT**	Trinité-et-Tobago
BJ	Bénin	**IE**	Irlande	**MN**	Mongolie	**UA**	Ukraine
BR	Brésil	**IL**	Israël	**MR**	Mauritanie	**UG**	Ouganda
BY	Bélarus	**IS**	Islande	**MW**	Malawi	**US**	Etats-Unis d'Amérique
CA	Canada	**IT**	Italie	**MX**	Mexique	**UZ**	Ouzbékistan
CF	République centrafricaine	**JP**	Japon	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Pays-Bas	**YU**	Yougoslavie
CH	Suisse	**KG**	Kirghizistan	**NO**	Norvège	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	République populaire démocratique de Corée	**NZ**	Nouvelle-Zélande		
CM	Cameroun			**PL**	Pologne		
CN	Chine	**KR**	République de Corée	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Roumanie		
CZ	République tchèque	**LC**	Sainte-Lucie	**RU**	Fédération de Russie		
DE	Allemagne	**LI**	Liechtenstein	**SD**	Soudan		
DK	Danemark	**LK**	Sri Lanka	**SE**	Suède		
EE	Estonie	**LR**	Libéria	**SG**	Singapour		

Procédé d'assistance à l'administration d'une application distribuée basée sur un fichier binaire de configuration dans un système informatique

La présente invention concerne un procédé d'assistance à l'administration d'une application distribuée basée sur un fichier binaire de configuration dans un système informatique. Ce procédé d'assistance à l'administration peut notamment être appliqué à un gestionnaire de traitement des transactions tel que celui commercialisé sous la marque "Tuxedo".

L'application "Tuxedo" permet à différents logiciels qui ne se connaissent pas mais qui respectent un certain protocole, de travailler ensemble.

Généralement, l'application "Tuxedo" est une application distribuée c'est-à-dire une application qui s'exécute sur plusieurs machines en même temps. On appelle "machine", le noeud du réseau au niveau duquel les serveurs de l'application "Tuxedo" s'exécutent, et "machine maître" celle contrôlant l'application "Tuxedo". La figure 8 illustre le fonctionnement de l'application "Tuxedo". Lorsque l'application "Tuxedo" est lancée, le fichier binaire de configuration (TUXCONFIG) est chargé du disque dans le tableau bulletin (Bulletin Board, BB) de la machine maître (Mm). Le tableau bulletin (BB) représente un ensemble de structures de données situées dans la mémoire partagée et contenant des informations sur les transactions, les serveurs, les services et les clients appartenant à l'application "Tuxedo". Lors du lancement de la machine maître (Mm), le tableau bulletin (BB) est chargé dans la mémoire de la machine maître (Mm) à partir du fichier binaire de configuration "Tuxedo" (TUXCONFIG). Puis, il est diffusé vers les machines esclaves (Me) par le processus maître de l'application appelé liaison distinguée du tableau bulletin DBBL (Distinguished Bulletin Board Liaison). Chaque machine de l'application est sous le contrôle d'un

processus appelé liaison du tableau bulletin BBL (Bulletin Bord Liaison). La liaison distinguée du tableau bulletin DBBL est un processus administratif qui communique avec les processus (BBL), pour coordonner les mises à jour du tableau bulletin (BB). La liaison du tableau bulletin BBL est un processus administratif chargé de tenir à jour une copie du tableau bulletin (BB) sur sa propre machine (Me). Chaque machine (Me) est sous le contrôle d'un processus appelé BBL, défini implicitement par "Tuxedo". Le pont (BRIDGE) (1) est un processus de gestion des communications entre les serveurs de l'application "Tuxedo". Chaque machine est dotée d'un pont défini implicitement par "Tuxedo". Le serveur TMS (Transaction Manager Server) est un processus qui gère un protocole de validation et la reprise pour les transactions exécutées par plusieurs serveurs applicatifs. Le module d'écoute (tlisten, 3) est un processus qui gère les messages destinés à l'application "Tuxedo" sur une machine donnée, avant que le processus pont (BRIDGE) de cette machine n'ait été lancée. Un module d'écoute permet à une machine de recevoir des informations provenant d'autres machines. Un module d'écoute est obligatoire sur chaque machine lorsque l'application est distribuée.

L'application "Tuxedo" est créée par la constitution d'un fichier binaire de configuration qui définit l'architecture de ladite application (figure 7). Lors de la création du fichier de configuration, un administrateur définit les services (Se) fournis par l'application et les assigne à des serveurs (Sr) d'application. L'administrateur définit ensuite des groupes (G) et assigne un ensemble de serveurs (Sr). Enfin, l'administrateur assigne des groupes (G) à une machine (M). Chaque application doit être dotée au minimum d'un groupe (G), d'un service (Se) et d'un serveur (Sr). Une machine (M) peut gérer plusieurs groupes (G).

Après la création d'une application "Tuxedo", celle-ci doit être administrée. L'objet de l'invention est de créer un système d'assistance à l'administration de l'application "Tuxedo". Les principales étapes concernant l'administration d'une application "Tuxedo" consistent en :

- une étape de chargement du fichier binaire de configuration de l'application "Tuxedo" ;

- une étape de lancement des modules d'écoute lorsque l'application "Tuxedo" est une application distribuée ;

- une étape de lancement de l'application Tuxedo ;

- une étape de contrôle de l'application. Celle-ci consiste à afficher des informations et à procéder, s'il y a lieu aux corrections requises ;

- une étape d'arrêt de l'application ; et éventuellement

- une étape d'arrêt des modules d'écoute lorsque ceux-ci ont été lancés.

L'administration d'une application distribuée peut rapidement devenir très complexe. En effet, avant que cette administration puisse démarrer, l'opérateur doit activer un module d'écoute sur chaque machine esclave sur laquelle il veut agir. Pour cela, l'administrateur doit tout d'abord consulter un fichier contenant des informations sur l'activation des modules d'écoute. Ce fichier est généralement stocké, à une place dont il faut se souvenir, sur chaque machine. Puis à l'aide de ces informations, l'opérateur doit activer tour à tour le module d'écoute de chaque machine. Ainsi, si l'application concerne dix machines, l'opérateur doit activer le module d'écoute sur les dix machines, puis à la fin de l'application, désactiver les dix modules d'écoute. Cette opération répétitive est longue et fastidieuse.

Pour effectuer ces tâches, chaque administrateur a sa solution. La solution la plus courante est de stocker sur chaque machine, a une place dont il faut se souvenir des scripts d'activation des modules d'écoute et d'avoir une copie papier du fichier de configuration. L'administrateur doit veiller à ce que les informations soient à jour à tout moment. A chaque fois que la configuration change, il ne doit pas oublier d'imprimer une copie papier du fichier de configuration et de mettre à jour les scripts sur les machines esclaves.

D'autre part, à chaque fois que l'opérateur veut agir sur un élément d'une application, il doit pouvoir identifier rapidement et de façon sûre une

ressource donnée comme par exemple, l'arrêt du serveur "serv1" appartenant au groupe "group1" sur la machine "mach1".

Lorsque le nombre d'application augmente, ces opérations manuelles sont source de nombreuses erreurs.

La présente invention a pour but de remédier aux inconvénients de l'art antérieur en proposant un procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions, basée sur le fichier binaire de configuration de l'application caractérisé en ce que ledit procédé comporte:

- une étape de décompilation du fichier de configuration actif de la machine maître,
- une étape de récupération d'informations dans le fichier de configuration décompilé de la machine maître (Mm),
- une étape de vérification de la consistance de ladite application mise en oeuvre sur une machine donnée.

Selon une autre particularité, ledit procédé permet de gérer au moins un module d'écoute (3) d'une machine quelconque de l'application à partir d'une autre machine.

Selon une autre particularité, les informations concernant ladite application distribuée sont directement prélevées dans le fichier de configuration actif de la machine maître.

Selon une autre particularité, l'étape de vérification de consistance de ladite application consiste en une comparaison entre des informations issues du fichier de configuration de la machine maître et des informations issues de ladite application courante mise en oeuvre sur une machine donnée.

Selon une autre particularité, ladite gestion des modules d'écoute permet de lancer et d'arrêter au moins un module d'écoute, d'afficher des informations concernant au moins un module d'écoute, de modifier le journal d'au moins un module d'écoute, de vérifier le script d'au moins un module d'écoute et de mettre à jour le script d'au moins un module d'écoute.

Selon une autre particularité, un administrateur se trouvant sur une machine quelconque du réseau peut lancer ou arrêter un module d'écoute mis en oeuvre sur une autre machine du réseau.

Selon une autre particularité, ledit procédé permet d'activer plusieurs modules d'écoute en une seule opération.

Selon une autre particularité, une interface graphique facilite la gestion des modules d'écoute.

Selon une autre particularité, ladite interface graphique permet de visualiser la structure de ladite application et de sélectionner une valeur voulue dans une liste de valeurs de la configuration courante.

Selon une autre particularité, lorsque le fichier contenant des informations sur ladite application mise en oeuvre sur une machine donnée (tlog) est inexistant, le procédé le génère automatiquement pour pouvoir l'utiliser lors du prochain lancement des modules d'écoute (3).

Selon une autre particularité, lesdites informations affichées concernant au moins un module d'écoute comprennent au moins le nom de ladite application, le nom logique de la machine (LMID) sur laquelle ladite application est exécutée, l'identification de l'administrateur (UID) de ladite application, l'adresse utilisé par le module d'écoute (NLSADRR), le chemin d'accès au réseau de ladite application, le chemin d'accès au fichier journal dudit module d'écoute (LLFPN).

D'autres particularités et avantages de la présente invention apparaîtront plus clairement à la lecture de la description ci-après faite en référence aux dessins annexés dans lesquels :

- la figure 1 représente une fenêtre de l'interface graphique proposant l'accès aux commandes principales de gestion des modules ;

- la figure 2 représente une fenêtre de l'interface graphique selon la figure 1 permettant d'activer un ou plusieurs modules d'écoute ;

- la figure 3 représente une fenêtre de l'interface graphique selon la figure 1 permettant l'arrêt d'un ou de plusieurs modules d'écoute ;

- la figure 4 représente une fenêtre de l'interface graphique selon la revendication 1 permettant l'affichage d'informations concernant un module d'écoute d'une application donnée ;

- la figure 5 représente une fenêtre de l'interface graphique selon la revendication 1 qui permet de vérifier le script d'un module d'écoute d'une application donnée ;

- la figure 6 représente une fenêtre de l'interface graphique selon la revendication 1 qui permet de mettre à jour le script d'un module d'écoute sur une machine donnée d'une application donnée ;

- la figure 7 représente la structure générale d'une application distribuée d'un gestionnaire de traitement des transactions ;

- la figure 8 représente un exemple d'application d'un gestionnaire de traitement des transactions.

Suit un exemple non limitatif de spécification de fichier de configuration. Ce fichier de configuration, présenté en annexe 1, concerne l'application "Tuxedo". Il est divisé en sept sections (ressources, machines, groupe, serveur, service, réseau).

La section ressource contient des informations générales concernant l'application. Ces informations sont communes à toutes les machines et sont constituées par les paramètres suivants :

- IPCKEY qui représente une clé numérique identifiant le segment de mémoire partagée dans lequel sont stockées les structures d'application. Grâce à cette clé numérique, une application donnée ne peut pas être en conflit avec d'autres applications ;

- MASTER qui représente la machine maître ;

- DOMAINID qui représente le domaine de l'application ;

- MAXACCESSERS qui définit le nombre maximum de personnes pouvant accéder à l'application ;

- MAXSERVERS qui définit le nombre maximum de serveurs pouvant être rattaché à l'application ;

- MAXSERVICES qui définit le nombre maximum de services pouvant être rattaché à l'application ;

- OPTIONS qui permet de préciser si l'application a lieu sur un réseau local ;

- MODEL qui permet de préciser si l'application est distribuée ou si elle ne l'est pas.

La section machines contient des informations sur chaque machine (puce, trifide, zig, orage) du réseau. Ces informations sont constituées par les paramètres suivants :

- LMID (Logital Machine ID) qui définit le nom logique de la machine c'est-à-dire le nom utilisé en interne par l'application, à la place du nom réseau;

- TUXDIR qui spécifie le chemin d'accès au répertoire d'installation du logiciel "Tuxedo" ;

- APPDIR qui spécifie le chemin d'accès aux serveurs applicatifs, c'est-à-dire le chemin menant aux programmes de l'application (par exemple les programmes concernant l'application "TUXEDO") ;

- TUXCONFIG qui spécifie le chemin d'accès absolu au fichier binaire de configuration TUXCONFIG, celui-ci contenant des informations sur l'application ;

- ENVFILE qui spécifie le chemin d'accès au fichier contenant les variables d'environnement pour les serveurs et pour les clients d'une machine donnée;

- ULOGPFX qui spécifie le chemin d'accès au fichier "ULOG" qui contient des informations sur l'historique de l'application.

La section groupe est la section dans laquelle chaque machine est attribuée à un groupe. Dans l'exemple de l'annexe 1, il existe quatre groupes. Un groupe est un ensemble de serveurs assurant des services

connexes. Dans le cas le plus simple, un groupe n'est constitué que d'un seul serveur. Tous les serveurs d'un groupe doivent s'exécuter sur la même machine. Une application doit comporter au moins un groupe.

La section serveur fournit des renseignements sur chaque serveur. Un serveur est un module fournisseur de services. Dans l'exemple à l'annexe 1, il existe quatre serveurs. Dans le cas le plus simple, un serveur assure un seul service. Une application doit être dotée d'au moins un serveur. La section serveur fournit les renseignements suivants :

- SRVGRP qui définit le groupe auquel le serveur est affilié ;
- SRVID qui définit le numéro d'identification du serveur ;
- MIN, MAX qui précisent le nombre maximum et minimum d'occurrences de ce serveur;
- RQADDR qui définit le nom de la queue de message utilisée pour l'envoi d'un message ;
- dans REPLYQ l'administrateur décide de l'existence d'une queue de réponse ;
- CLOPT qui indique les options de démarrage du serveur (services disponibles priorité, ...).

Dans la section service, l'administrateur peut spécifier les services. Un service est un ensemble de fonctions répondant à des requêtes de services émanant d'utilisateurs finaux de l'application. Si l'administrateur désire indiquer des valeurs facultatives différentes des valeurs par défaut, les services doivent obligatoirement être définis.

La section réseau (network) contient pour chaque machine :

- l'adresse complète utilisée par le processus pont (BRIDGE) appelée "Network Address" ou "NADDR". Les quatre premiers chiffres (0002 dans l'exemple de la figure 4) représentent le protocole de communication utilisé ("tcp" dans l'exemple ci-dessus). Les quatre chiffres suivants représentent le numéro de port utilisé par le processus et les chiffres suivants représentent l'adresse réseau de la machine ;

- le chemin d'accès au pont (BRIDGE) de la machine. Le pont est un processus de gestion des communications entre les serveurs de l'application. Il sert à amorcer l'application. Chaque machine est dotée d'un pont.

- l'adresse complète du module d'écoute appelée "NLSADDR". Les quatre premiers chiffres représentent le protocole de communication utilisé. Les quatre chiffres suivants représentent le numéro de port utilisé par le module d'écoute qui doit être différent de celui utilisé par le processus pont (BRIDGE). Les chiffres suivants représentent l'adresse réseau de la machine.

La particularité de l'invention est que les informations concernant l'application sont directement prélevées dans le fichier actif de la machine maître. Un administrateur se trouvant sur une machine quelconque du réseau peut gérer l'exécution de la commande "get_tuxval" sur la machine maître pour le compte de l'administrateur comme représenté en page 27 de l'annexe 2.

La sous routine "get_tuxconfig" du programme utilisé dans la mise en oeuvre du procédé d'assistance à l'administration d'une application distribuée, recherche sur le disque dur de la machine maître le fichier actif de configuration de l'application. Celui-ci est ensuite décompilé au moyen de la commande "tmunloadcf" (Page 28 de Annexe 2, Lignes 85 à 99).

```
get_tuxconfig() {
        if [ -s tuxconf.tmp.$appname ]
        then
                cat tuxconf.tmp.$appname
        else
                rm  -f tuxconf.tmp.*
                prog="$Env"'
$TUXDIR/bin/tmunloadcf
echo "\nexit $?"
'
#print -r "$prog" > prog
                rsh "$MASTER" -l "$ADMIN" "$prog" | tee tuxconf.tmp.$appname
        fi
get_tlistenlog
```

```
}
```

La sous routine "get_tuxval" de ce programme (Page 28 de l'annexe 2, lignes 112 à 183) prélève les paramètres tels que LMID, APPDIR, TUXCONFIG, TUXDIR, ROOTDIR, ULOGPFX, NLSADDR, UID et BRIDGE du fichier binaire de configuration de l'application obtenue à l'aide de la sous routine "get_tuxconfig".

```
get_tuxval() {
        get_tuxconfig | \
        sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\V0/g' | awk '
```

Les valeurs des paramètres recherchées sont tout d'abord initialisées. Pour cela des matrices associatives appelées "tuxconfig_section" sont créées.

```
BEGIN {
   tuxconfig_section["*RESOURCES"] = 1
   tuxconfig_section["*MACHINES"] = 2
   tuxconfig_section["*GROUPS"] = 3
   tuxconfig_section["*SERVERS"] = 4
   tuxconfig_section["*SERVICES"] = 5
   tuxconfig_section["*ROUTING"] = 6
   tuxconfig_section["*NETWORK"] = 7
}
```

Un index est associé à chaque matrice. Les paramètres recherchés sont situés dans différentes sections du fichier de configuration. Par exemple pour l'application "Tuxedo", ces différentes sections, au nombre de sept, sont appelées "Ressources", "Machines", "Groupes", "Serveurs", "Services" et "Réseau". Pour pouvoir prélever les paramètres dont l'ordinateur a besoin, il doit pouvoir repérer l'endroit où il se trouve dans le fichier de configuration. Dans ce programme, lorsque le nombre de champ (NF) est égal à 1, l'ordinateur se trouve au début d'une section.

```
NF == 1 {
   if ( $1 in tuxconfig_section ) {
   section = tuxconfig_section[$1]
```

```
next
   }
  }
```

Si l'ordinateur est dans la section 2 et que le deuxième mot est LMID, l'ordinateur prélève le nom logique de la machine (LMID) sur laquelle l'administrateur se trouve.

```
section == 2 && $2 == "LMID" {  # MACHINES section
 if ( $3 == machine) {
  printf "uname=%s\n", $1
  mach_found=1
 }
 else { # reset mach_found for furtheur machines
  mach_found = 0
 }
 next
}
```

Si l'ordinateur est dans la section 2 et que le premier mot est APPDIR, il prélève le chemin d'accès au répertoire sous lequel les serveurs sont amorcés.

```
section == 2 && $1=="APPDIR" && mach_found==1 {
                printf "appdir=%s\n", $2
   appdir = $2
   next
        }
```

En procédant de la même manière, l'ordinateur va relever successivement dans la section machine du fichier de configuration le chemin d'accès absolu au fichier binaire de configuration (TUXCONFIG), le chemin d'accès au répertoire d'installation du logiciel Tuxedo (TUXDIR ou ROOTDIR), des informations sur l'historique de l'application (ULOGPFX) et dans la section réseau l'adresse du pont de la machine (NLSADDR).

```
section == 2 && $1=="TUXCONFIG" && mach_found == 1 {
                printf "tuxconfig=%s\n", $2
```

```
    next
        }
section == 2 && $1=="TUXDIR" && mach_found==1 {
        printf "tuxdir=%s\n", $2
  next
        }
section == 2 && $1=="ROOTDIR" && mach_found==1 { # for V4
            printf "tuxdir=%s\n", $2
    next
        }
section == 2 && $1=="ULOGPFX" && mach_found==1 {
    ulogpfx=1; printf "ulogpfx=%s\n", $2
    next
    }
section == 7 && NF == 1 {
  if ( $1 == machine )
    {mach_found = 1}
  else { # reset mach_found for other machines
    mach_found = 0
    }
  next
  }
section == 7 && $1=="NLSADDR" && mach_found==1 {
        printf "nlsaddr=%s\n", $2
  next
        }
```

Le programme exécute une boucle sur cette sous routine pour chaque machine jusqu'à ce que l'ordinateur trouve la machine courante. Puis, l'ordinateur se procure dans la section ressources du fichier de configuration l'identification de l'utilisateur de l'application (UID).

```
section == 1 && $1 == "UID" {printf "uid=%s\n", $2 ;next }
```

Si aucune valeur n'a été définie pour l'UID dans le fichier de configuration, c'est l'UID de la personne qui a construit l'application qui sera utilisé. Puis, l'ordinateur relève dans la section réseau du fichier de configuration le chemin d'accès au pont (BRIDGE) de la machine.

```
section == 7 &&      $1=="BRIDGE" && mach_found==1 {
```

Le paramètre ULOGPFX représentant l'historique de la machine est une valeur optionnelle. Lorsqu'il est inexistant, l'ordinateur va générer un fichier appelé "ULOG" dans le répertoire APPDIR contenant des informations sur les manipulations opérées sur l'application.

```
if ( ulogpfx == 0 ) {
   printf "ulogpfx=%s/ULOG\n", appdir }
        } ' machine=$machine appname=$appname
    lang=`sed -e "s/=/ /g" -e "s/'//g" -e "s/;/ /" $ConfDir/$appname.tux | awk '
      $1 == "LANG" {printf "lang=", $2}' `
   }
```

De plus, l'ordinateur a besoin de la langue de travail de l'application représentée par le paramètre LANG, ainsi que de la valeur "tlog". Le paramètre LANG se trouve dans le fichier de configuration de l'utilisateur.

```
lang=`sed -e "s/=/ /g" -e "s/'//g" -e "s/;/ /" $ConfDir/$appname.tux | awk '
      $1 == "LANG" {printf "lang=", $2}' `
```

La valeur "tlog" fait référence au fichier "tlistenlog . <nom de l'application> . <nom de la machine>" contenant le nom du fichier historique du module d'écoute.

Dans la sous routine get_tuxval, le programme a rassemblé toutes les variables d'environnement dont il a besoin pour pouvoir lancer le procédé d'assistance à l'administration d'une application distribuée. Ce procédé permet, en outre d'amorcer et d'arrêter un ou plusieurs modules d'écoute, d'afficher des informations sur un ou plusieurs modules d'écoute, de modifier le journal d'un ou plusieurs modules d'écoute, de vérifier le script d'un ou plusieurs modules d'écoute et enfin de mettre à jour le script d'un ou plusieurs modules d'écoute (Figure 1).

Le procédé d'assistance à l'administration d'une application "Tuxedo" distribuée est doté d'une interface graphique qui permet l'accès aux commandes du gestionnaire de traitement des transactions. Pour exécuter une tâche, l'administrateur n'est pas tenu d'entrer des commandes,

il lui suffit de cliquer sur des icônes, d'appeler des menus et de spécifier des valeurs via des boîtes de dialogue. Le procédé d'assistance est piloté par menus, structurés sous forme d'arborescence. La sélection d'une option dans le menu principal entraîne l'affichage du menu de niveau inférieur associé. Ce processus est répété jusqu'à l'affichage d'une boite de dialogue déroulante dans laquelle l'administrateur doit entrer des valeurs de paramètre. Afin de pouvoir gérer les modules d'écoute de l'application "Tuxedo" distribuée, l'administrateur sélectionne à partir du menu principal de "Tuxedo Commands", les fonctions "Tuxedo Commands", "Start/Stop Tuxedo Configuration", "Set up a Tuxedo Application" et "Manage the Listener Processes". Les fonctions sélectionnables "Strart Listener Processes ", "Stop Listener Processes", "Change/Schow Listener Process Parameters", "Schow currently running Listener Processes", "Check consistency of Listener Process scripts with TUXCONFIG Level" et "Update Listener Process to TUXCONFIG Level" apparaissent sur la fenêtre de l'interface graphique (Figure 1). Pour lancer des modules d'écoute, l'administrateur doit sélectionner la commande "Start Listener Processes" en positionnant le curseur de sa souris sur le pavé (11) et en appuyant sur le bouton gauche de sa souris. La fenêtre de la figure 2 apparaît après la sélection. Si une application a été préalablement désignée, son nom est affiché sur le pavé (21). Sinon, l'administrateur est informé par la marque clignotante du curseur qu'il doit en donner une. Pour cela, l'administrateur peut soit cliquer sur le bouton "List" (22) pour afficher la liste des applications enregistrées et en sélectionner une, soit entrer explicitement le nom de l'application désirée. Puis l'administrateur est informé par la marque clignotante du curseur dans le pavé (23), à partir de laquelle il doit préciser le nom des machines sur lesquelles un module d'écoute doit être lancé. De la même façon, la liste des machines comprises dans ladite application peut ˆtre obtenue en cliquant sur le bouton "List" (22). Pour valider les machines sélectionnées, par exemple par surbrillance, l'administrateur doit cliquer sur le bouton "OK" (24). La commande de démarrage du module d'écoute est

obtenue par la sélection du bouton "Command" (25). Le bouton "Reset" (26) permet de réinitialiser les valeurs des pavés (21) et (23). Le bouton "Cancel" (27) permet d'annuler une valeur introduite sur les pavés (21) et (23). Le bouton "?" (28) offre une aide en ligne à l'administrateur.

Pour chaque machine désignée dans la liste des machines, l'ordinateur se procure des informations sur l'application dans le fichier de configuration de la machine maître et un fichier historique appelé fichier "tlistenlog . <nom de l'application> . <nom de la machine>" contenant des informations sur l'application agissant actuellement sur cette machine. L'ordinateur vérifie d'abord si le module d'écoute n'est pas déjà démarré sur la machine. Si c'est le cas, le message "Listener already running on <nom de la machine>" est imprimé sur l'écran. Sinon, si un fichier local existe, l'ordinateur l'exécute et imprime le message "Listener started on the machine", si la commande réussit. Si la commande échoue, l'ordinateur imprime le message "Listener starting failed on <nom de la machine>". Si le fichier local n'existe pas, l'ordinateur génère un fichier "tlistenlog . <nom de l'application> . <nom de la machine>" dans le répertoire APPDIR, l'exécute et rend compte du résultat comme précédemment Ce fichier contient des informations sur l'application courante et sera utilisé dans le prochain lancement des modules d'écoute. Ceci correspond aux lignes 652 à 698 de la page 36 et aux lignes 699 à 719 de la page 37 de l'annexe 2.

```
startlistproc)
appname=$1; shift
                list="$*"
                set_environ
                boucle_status=0
                exit_status=0
                for machine in $list
                do
                  echo "\n------- Machine: $machine -------\n"
                  get_tuxval > "appname.tux"
              get_tllog
              . ./appname.tux
              prog1="
              TUXDIR=$tuxdir; export TUXDIR
```

```
                ROOTDIR=$tuxdir; export ROOTDIR  # V4
            APPDIR=$appdir; export APPDIR
            TUXCONFIG=$tuxconfig; export TUXCONFIG
            PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
            LANG=$lang; export LANG
            LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
            COLUMNS=200; export COLUMNS
            ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {exit 1}'
            if [ \$? = 1 ]
             then
               echo \"Listener already running on $machine\"
               echo exit 0
                 exit 0
                      fi
            if [ -f $appdir/tlisten.$appname.$machine ]
            then
                . $appdir/tlisten.$appname.$machine
                      ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~
\"$nlsaddr\" {exit 1}'
                      if [ \$? = 1 ]
                      then
                            echo \"Listener started on $machine\"
                            echo exit 0
                      else
                        echo \"Listener starting failed on $machine !!!\"
                            echo exit 1
                      fi
             else # create the script file & exec it
                echo \"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L $tllog\" >
$appdir/tlisten.$appname.$machine
                chmod ug+x $appdir/tlisten.$appname.$machine
                . $appdir/tlisten.$appname.$machine
                ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {exit
1}'
                if [ \$? = 1 ]
                 then
                  echo \"Listener started on $machine\"
                  echo exit 0
                 else
                    echo \"Listener starting failed on $machine !!!\"
                    echo exit 1
                 fi
            fi"
        #echo "$prog1" > prog1
        if [ -z "$uname" ]
          then
            print "Host $machine not found"
            exit 1
```

```
    fi
      rsh "$uname" -l "$ADMIN" "$prog1" | awk '
            NR == 1 {line = $0}
            NR > 1 { print line; line = $0 }
            END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
      boucle_status=`expr $boucle_status \| $? `
      done
       exit $boucle_status
    ;;
```

Pour arrêter un module d'écoute, l'administrateur sélectionne à partir du menu principal de gestion des modules d'écoute "Manage the Listener Processes", la fonction "Stop Listener Processes" en positionnant son curseur sur la pavé (12) (Figure 1). La fenêtre de la figure 3 apparaît. Elle permet d'indiquer dans un premier pavé (31), le nom de l'application, dans un second pavé (32), le nom de la ou des machines. En cliquant sur le bouton "List" (33), une liste des applications enregistrées ou une liste des machines concernant chaque application peut être obtenue selon la position de la marque de position clignotante (34). Pour chaque machine de l'application, l'ordinateur imprime le nom de la machine pour laquelle le module d'écoute est arrêté. Cette sélection à l'écran grâce à l'interface graphique lance les pas de programmes "stoplistproc" au cours desquels le programme procure à la station sur laquelle la procédure d'arrêt est lancée, des informations par get_tuxval sur l'application, contenue dans le fichier de configuration de la machine maître (Page 37 de l'Annexe 2, Lignes 720 à 762).

```
stoplistproc)
        appname=$1; shift
        list="$*"
        set_environ
        boucle_status=0
        exit_status=0
        for machine in $list
       do
          echo "\n------- Machine: $machine -------\n"
        get_tuxval > "appname.tux"
        . ./appname.tux
```

```
        prog1="
          COLUMNS=200; export COLUMNS
          ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {print \$2;
    exit 0 }' | read pid
          if [ -n \"\$pid\" ]
            then
                kill -9 \$pid > /dev/null
                status=\$?
                if [ \$status -eq 0 ]
                then
                        echo \"Process \$pid killed on $machine\"
                        echo exit 0
                else
                        echo \"Failed to stop listener on $machine!!!\"
                        echo exit 1
                fi
          else
                echo \"No Listener running on $machine\"
                echo exit 1
          fi"
        if [ -z "$uname" ]
          then
            print "Host $machine not found"
            exit 1
        fi
          rsh "$uname" -l "$ADMIN" "$prog1" | awk '
                NR == 1 {line = $0}
                NR > 1 { print line; line = $0 }
                END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
          boucle_status=`expr $boucle_status \| $? `
        done
      exit $boucle_status
    ;;
```

Si un processus appelé "tlisten" appartenant à l'application courante est en fonctionnement sur cette machine, l'ordinateur l'arrête (kill) et imprime le message "Process <l'identification du process (PID, Process IDentifier)> killed on <nom de la machine>", sinon il imprime le message "Failed to stop listener on <nom de la machine>".

De plus, ce procédé d'assistance à l'administration d'une application permet d'afficher des informations concernant un module d'écoute. Pour cela à partir du menu principal de gestion des modules d'écoute "Manage the

Listener Processes", il suffit à l'administrateur de sélectionner la fonction "Change/Show Listener Processes Parameters" sur le pavé (13) de la fenêtre présentée en Figure 1. La fenêtre de la figure 4 apparaît. L'administrateur doit préciser dans le pavé (41), le nom de l'application et dans le pavé (42), un nom de machine. Suite à cette précision, les autres pavés (43 à 46) de la fenêtre font apparaître les valeurs des paramètres tels que :

- l'identification de l'administrateur (UID),
- l'adresse complète du module d'écoute composée de l'adresse de la machine et du numéro de port qu'il utilise (NLSADRR),
- le chemin d'accès au réseau,
- le chemin d'accès complet au fichier journal du module d'écoute (Listener Logfile Full Path Name, LLFPN),

Toutes ces informations sont extraites du fichier TUXCONFIG de la machine maître. Ces informations ne sont pas modifiables par cette commande, à l'exception du LLFPN. L'annexe 2 présente aux lignes 570 à 579 de la page 35, la partie du programme correspondant à l'exécution de la commande de modification du LLFPN.

```
chglisten)
        appname=$1
        machine=$2
        shift 2
        if [ $# -gt 0 ]
         then
          echo "TLLOG $machine $1" > $ConfDir/tlistenlog.$appname.$machine
        fi
        exit $?
            ;
            ;
```

Pour pouvoir visualiser les modules d'écoute actifs de l'application, l'administrateur doit sélectionner la fonction "Show currently running Listener Processes" en cliquant sur le pavé (14) de la fenêtre de la Figure 1. L'ordinateur affiche la liste des machines de l'application sur lesquelles un module d'écoute est actif et l'identification du processus PID (Process

Identifier) appartenant à la configuration du réseau. L'annexe 2 présente aux lignes 764 à 768 de la page 37 et aux lignes 769 à 809 de la page 38, la partie de programme correspondant à l'affichage de la liste des modules d'écoute actifs, qui utilise la fonction get_tuxval.

```
runninglist)
        appname=$1
        boucle_status=0
        set_environ
        list_lmids=`get_tuxconfig | \
        sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\V0/' -e "s/\*//"  | awk '
          BEGIN { network=0 }
          {line = $0}
          NF == 1 { if (network == 1) print $1}
          $1 == "NETWORK" { network = 1}
          END {if(sub("^exit ","", line)) exit line; exit -1 }' `
        for machine in $list_lmids
         do
          get_tuxval > "appname.tux"
          . ./appname.tux
          prog1="
          TUXDIR=$tuxdir; export TUXDIR
          LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
          ROOTDIR=$tuxdir; export ROOTDIR  # V4
          APPDIR=$appdir; export APPDIR
          TUXCONFIG=$tuxconfig; export TUXCONFIG
          PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
          LANG=$lang; export LANG
          COLUMNS=200; export COLUMNS
          ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {print \$2}' |
read pid
          if [ -n \"\$pid\" ]
           then
            echo \"Listener running on $machine: pid = \$pid\"
            echo exit 0
           else
            echo \"No Listener running on $machine\"
            echo exit 0
          fi"
          if [ -z "$uname" ]
           then
            print "Host $machine not found"
            exit 1
          fi
          rsh "$uname" -l "$ADMIN" "$prog1" | awk '
```

```
        NR == 1 {line = $0}
        NR > 1  { print line; line = $0}
        END { if (sub("^exit ", "", line)) exit line; print line; exit -1} '
        boucle_status=`expr $boucle_status \| $?`
      done
      exit $boucle_status
    ;;
```

L'administrateur peut aussi vérifier le script d'un module d'écoute. En sélectionnant la fonction "Check consistency of Listener Process scripts with Tuxconfig" sur le pavé (15) de la fenêtre représentée en figure 1, la fenêtre de la figure 5 apparaît. L'administrateur doit entrer le nom d'une application sur le pavé (51) et le nom d'une machine donnée sur le pavé (52). Une liste des applications et des machines est à la disposition de l'administrateur grâce au bouton "List" (53). Le programme compare les informations contenues dans le fichier TUXCONFIG de la machine maître et extraites par la fonction "get_tuxval" avec les informations contenues dans le fichier "tlisten.(nom de l'application).(nom de la machine)" situé dans le répertoire APPDIR de la machine et donne le résultat de cette comparaison. L'annexe 2 présente aux lignes 580 à 631 de la page 35 et aux lignes 632 à 651 de la page 36, la partie du programme correspondant à la vérification d'un script d'un module d'écoute qui permet de signaler les discordances entres les paramètres des fichiers en imprimant par exemple pour le pont "BRIDGE values mismatch".

```
chklistscript)
        appname=$1
      machine=$2
      set_environ
      get_tuxval > "appname.tux"
      get_tllog
      . ./appname.tux
      prog="
      if [ -f $appdir/tlisten.$appname.$machine ]
       then
        cat $appdir/tlisten.$appname.$machine
        echo \"\\nexit 0\"
       else
```

```
  echo \"\\nexit 1\"
fi"
if [ -z "$uname" ]
 then
   print "Host $machine not found"
   exit 1
fi
rm -f tlscript.$appname.$machine
rsh "$uname" -l "$ADMIN" "$prog" | tee tlscript.$appname.$machine >
/dev/null
[ $? -ne 0 ] && exit 1
[ -s tlscript.$appname.$machine ] && cat tlscript.$appname.$machine |
awk '
 END { if ( $2 == "1" ) exit -1} '
[ $? -eq -1 ] && exit 1
[ -s tlscript.$appname.$machine ] && cat tlscript.$appname.$machine | \
awk '
 $1 ~ "tlisten" {
   mismatch = 0
   fexec=sprintf("%s/bin/tlisten", tuxdir)
   if ($1 != fexec) {
     print "tlisten command full pathnames mismatch"
     printf "\tscript:\t%s\n", $1
     printf "\tconfig:\t%s\n", fexec
     mismatch +=1
     }
   for (i=2; i <= NF; i++) {
     if (( $i == "-d") && ($(i+1) != bridge)) {
       print "BRIDGE values mismatch"
       printf "\tscript:\t%s\n",$(i+1)
       printf "\tconfig:\t%s\n",bridge
       mismatch +=1
       }
     if (( $i == "-l") && ($(i+1) !=nlsaddr)) {
       print "NLSADDR values mismatch"
       printf "\tscript:\t%s\n",$(i+1)
       printf "\tconfig:\t%s\n", nlsaddr
       mismatch +=1
       }
     if (( $i == "-u") && ($(i+1) != uid)) {
       print "UID values mismatch"
       printf "\tscript:\t%s\n", $(i+1)
```

```
                printf "\tconfig:\t%s\n", tllog
                mismatch +=1
                }
              }}
        END {
          if ( mismatch == 0 )
           printf "Script File is up-to-date for %s\n",machine
          else
           printf "\nScript File is NOT up-to-date for %s\n",machine
          } ' tllog=$tllog machine=$machine bridge=$bridge \
            nlsaddr=$nlsaddr uid=$uid tuxdir=$tuxdir
exit $?
;;
```

Un script d'un module d'écoute peut aussi être mis à jour par la sélection de la fonction "Update Listener Process scripts to TUXCONFIG Level". Un script d'un module d'écoute Tuxedo permet de lancer un module d'écoute. Il suffit d'intégrer un script de ce type pour une machine donnée, dans la séquence de lancement pour que le module d'écoute soit lancé automatiquement en même temps que la machine. Dans la fenêtre représenté figure 6, l'administrateur entre sur le pavé (61) le nom d'une application, et sur le pavé (62) le nom d'une ou de plusieurs machines. Le programme se procure par l'appel de la sous routine "get_tuxval", toutes les informations dont il a besoin dans le fichier binaire de configuration extraites par la sous routine "get_tuxconfig" et crée un fichier lui correspondant dans le répertoire APPDIR sous le nom "tlisten.(nom de l'application).(nom de la machine). Les lignes 810 à 831 de l'annexe 2 page 38 présente la partie du programme correspondant à l'exécution de la commande de mise à jour d'un script d'un module d'écoute.

```
updtlistscript)
      appname=$1
      machine=$2
      set_environ
      get_tllog
      get_tuxval > "appname.tux"
      . ./appname.tux
      prog="
```

```
echo \"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L $tllog\" >  $app
dir/tlisten.$appname.$machine
        chmod ug+x $appdir/tlisten.$appname.$machine
        echo exit \$?"
        if [ -z "$uname" ]
          then
            print "Host $machine not found"
            exit 1
        fi
        rsh "$uname" -l "$ADMIN" "$prog" | awk '
          NR == 1 {line = $0}
          NR > 1 { print line; line = $0 }
          END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
        exit $?
        ;;
```

D'autres modifications à la portée de l'homme de métier font également partie de l'esprit de l'invention.

ANNEXE 1

Nov 20 1997 16:23:57 **ubb.dom1** Page 1

```
1     #
2     #       Tuxedo configuration UBBCONFIG for the model TEST1
3     #
4
5     *RESOURCES
6     IPCKEY                  191785
7     MASTER                  site1
8     DOMAINID                        dom1
9     MAXACCESSERS    50
10    MAXSERVERS              50
11    MAXSERVICES             100
12    OPTIONS                 LAN
13    MODEL                   MP
14
15    *MACHINES
16    puce            LMID=site1
17                    TUXDIR="usr/tuxedo"
18                    APPDIR="/home/dia/tuxedo"
19                    TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
20                    ENVFILE="/home/dia/tuxedo/envfile_puce"
21                    ULOGPFX="/home/dia/tuxedo/ULOG"
22
23    trifide         LMID=site2
24                    TUXDIR="usr/tuxedo"
25                    APPDIR="/home/dia/tmp"
26                    TUXCONFIG="/home/dia/tmp/TUXCONFIG"
27                    ENVFILE="/home/dia/tmp/envfile_trifide"
28                    ULOGPFX="/home/dia/tmp/ULOG"
29
30    zig             LMID=site3
31                    TUXDIR="usr/tuxedo"
32                    APPDIR="/home/dia/tuxedo"
33                    TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
34                    ENVFILE="/home/dia/tuxedo/envfile_zig"
35                    ULOGPFX="/home/dia/tuxedo/ULOG"
36
37    orage           LMID=site4
38                    TUXDIR="usr/tuxedo"
39                    APPDIR="/home/dia/tuxedo"
40                    TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
41                    ENVFILE="/home/dia/tuxedo/envfile_orage"
42                    ULOGPFX="/home/dia/tuxedo/ULOG"
43
44
45
46    *GROUPS
47
48    DEFAULT:        TMSNAME=TMS     TMSCOUNT=2
49    GROUP1          LMID=site1
50                    GRPNO=1
51    GROUP2          LMID=site2
52                    GRPNO=2
53    GROUP4          LMID=site3
54                    GRPNO=3
55    GROUP3          LMID=site4
56                    GRPNO=4
57
58
59    *SERVERS
60    #
61    DEFAULT: RESTART=Y MAXGEN=5 REPLYQ=Y CLOPT="-A"
62
63    SRV1
64                    SRVGRP=GROUP1
65                    SRVID=100
66                    MIN=2   MAX=2
67                    RQADDR=QSRV1_1
68                    REPLYQ=Y
69                    CLOPT="-s SVC1_1 -s SVC1_2 -- "
70    SRV2
71                    SRVGRP=GROUP2
```

ANNEXE 1

Nov 20 1997 16:23:57 **ubb.dom1** Page 26

```
72                      SRVID=200
73                      MIN=2    MAX=2
74                      RQADDR=QSRV2_2
75                      REPLYQ=Y
76                      CLOPT="-s SVC2_1 -s SVC2_2 -- "
77      SRV4
78                      SRVGRP=GROUP4
79                      SRVID=300
80                      MIN=2    MAX=2
81                      RQADDR=QSRV4_3
82                      REPLYQ=Y
83                      CLOPT="-s SVC4_1 -s SVC4_2 -- "
84      SRV3
85                      SRVGRP=GROUP3
86                      SRVID=400
87                      MIN=2    MAX=2
88                      RQADDR=QSRV3_4
89                      REPLYQ=Y
90                      CLOPT="-s SVC3_1 -- "
91
92
93      *SERVICES
94      DEFAULT:        LOAD=50
95      SVC1_1
96      SVC1_2
97      SVC2_1
98      SVC2_2
99      SVC4_1
100     SVC4_2
101     SVC3_1
102
103
104
105     *NETWORK
106     site1
107     #       port number=60951 (ee17 hexa)
108     #       local address=81b683e0
109             NADDR="\\x0002ee1781b683e00000000000000000"
110             BRIDGE="/dev/xti/tcp"
111     #       port number=60952 (ee18 hexa)
112             NLSADDR="\\x0002ee1881b683e00000000000000000"
113     #
114     site2
115     #       port number=60951 (ee17 hexa)
116     #       local address=81b68387
117             NADDR="\\x0002ee1781b6838700000000000000000"
118             BRIDGE="/dev/xti/tcp"
119     #       port number=60952 (ee18 hexa)
120             NLSADDR="\\x0002ee1881b6838700000000000000000"
121     #
122     site3
123     #       port number=60951 (ee17 hexa)
124     #       local address=81b683e1
125             NADDR="\\x0002ee1781b683e10000000000000000"
126             BRIDGE="/dev/xti/tcp"
127     #       port number=60952 (ee18 hexa)
128             NLSADDR="\\x0002ee1881b683e10000000000000000"
129     #
130     site4
131     #       port number=60951 (ee17 hexa)
132     #       local address=81b6838b
133             NADDR="\\x0002ee1781b6838b0000000000000000"
134             BRIDGE="/dev/xti/tcp"
135     #       port number=60952 (ee18 hexa)
136             NLSADDR="\\x0002ee1881b6838b0000000000000000"
137     #
138
```

ANNEXE 2

```
1
2     # @BULL_COPYRIGHT@
3     #
4     # HISTORY
5     # $Log: smtuxadmin.ksh,v $
6     # Revision 1.7  1996/02/12  11:40:49  odeadm
7     #      bci V1Set2C 23.01.96
8     #      [1996/01/23  14:31:07  dia]
9     #
10    # Revision 1.6  1995/12/20  14:26:59  odeadm
11    #      V1 Set2: Still troubles with smtuxadmin.ksh
12    #      [1995/12/11  11:56:55  odeadm]
13    #
14    #      07.12.95 V1Set2 first batch of corrections
15    #      [1995/12/07  17:22:57  odeadm]
16    #
17    #      *** empty log message ***
18    #      [1995/11/30  13:48:30  dia]
19    #
20    #      *** empty log message ***
21    #      [1995/11/30  13:48:30  dia]
22    #
23    # Revision 1.5  1995/10/13  11:52:51  odeadm
24    #      Servers TMS/Partitioned mach.
25    #      [1995/10/09  12:05:57  dia]
26    #
27    # Revision 1.4  1995/09/15  15:15:06  odeadm
28    #      Corrections MRs BUILD 3
29    #      [1995/09/07  15:45:27  dia]
30    #
31    # Revision 1.3  1995/08/24  13:38:03  odeadm
32    #      Build3
33    #      [1995/08/23  09:04:31  odeadm]
34    #
35    # Revision 1.2  1995/07/19  15:18:13  odeadm
36    #      Madison build M0.2
37    #      [1995/07/10  10:01:58  odeadm]
38    #
39    # $EndLog$
40    #! /bin/ksh
41    ConfDir=$WRAPPING_CONFIGURATION
42    Context=smtuxedo.ctx
43    Scanconf=$MADISON_VAR/surveyor/scanconf.tux
44    V5_to_V4='ROOTDIR=$TUXDIR; export ROOTDIR'
45    Set1_to_Set2='[ -z "$ADMIN" ] && export ADMIN="madison"'
46    cmd=$1; shift
47
48    set_environ() {
49          MASTER=""; APPDIR=""; ADMIN=""
50          filename=$ConfDir/$appname.tux
51    Env=`tuxgetenv -k -v APP_PW $filename << !
52    tuxgetenvp
53    !`
54          eval "$Env"; unset APP_PW
55          eval "$Set1_to_Set2"
56          if [ -n "$MASTER" -a -n "$APPDIR" ]
57          then
58                Env="$Env
59    $PW
60    $Set1_to_Set2
61    $V5_to_V4"'
62    LD_LIBRARY_PATH=$LIBPATH; export LD_LIBRARY_PATH;
63    cd $APPDIR
64    PATH=${PATH}:.:$APPDIR:$TUXDIR/bin; export PATH'
65                return 0
66          fi
67          exit 1
68    }
69
70    remote_cmd() {
71          prog="$Env
```

ANNEXE 2

```
$cmd"'
status=$?
sleep 1
echo "\nexit $status"
'
#print -r "$prog" > prog
       rsh "$MASTER" -l "$ADMIN" "$prog" | awk '
              NR == 1 {line = $0}
    NR > 1  { print line; line = $0}
              END {if(sub("^exit ","", line)) exit line; exit -1 }'
}


get_tuxconfig() {
       if [ -s tuxconf.tmp.$appname ]
       then
              cat tuxconf.tmp.$appname
       else
              rm  -f tuxconf.tmp.*
              prog="$Env"'
$TUXDIR/bin/tmunloadcf
echo "\nexit $?"
'
#print -r "$prog" > prog
              rsh "$MASTER" -l "$ADMIN" "$prog" | tee tuxconf.tmp.$appname
       fi
get_tlistenlog
}

get_tlistenlog() {
       tllogfname=$ConfDir/tlistenlog.$appname.$machine
if [ -s $tllogfname ]
 then
   cat $tllogfname
 else # default value
   echo "TLLOG $machine $MADISON_TMP/tlisten.$appname.$machine.log" | tee $tllogfname
fi
   echo "\nexit $?"
}

get_tuxval() {
       get_tuxconfig | \
       sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\\/0/g' | awk '
  BEGIN {
    tuxconfig_section["*RESOURCES"] = 1
    tuxconfig_section["*MACHINES"]  = 2
    tuxconfig_section["*GROUPS"]  = 3
    tuxconfig_section["*SERVERS"] = 4
    tuxconfig_section["*SERVICES"]  = 5
    tuxconfig_section["*ROUTING"] = 6
    tuxconfig_section["*NETWORK"] = 7
}
 NF == 1 {
    if ( $1 in tuxconfig_section ) {
    section = tuxconfig_section[$1]
    next
    }
  }
 section == 2 && $2 == "LMID" {  # MACHINES section
 if ( $3 == machine) {
   printf "uname=%s\n", $1
   mach_found=1
  }
  else { # reset mach_found for furtheur machines
   mach_found = 0
  }
  next
 }
section == 2 && $1=="APPDIR" && mach_found==1 {
              printf "appdir=%s\n", $2
    appdir = $2
```

ANNEXE 2

```
143       next
144          }
145   section == 2 && $1=="TUXCONFIG" && mach_found == 1 {
146                printf "tuxconfig=%s\n", $2
147       next
148          }
149   section == 2 && $1=="TUXDIR" && mach_found==1 {
150         printf "tuxdir=%s\n", $2
151     next
152          }
153   section == 2 && $1=="ROOTDIR" && mach_found==1 { # for V4
154                printf "tuxdir=%s\n", $2
155       next
156          }
157   section == 2 && $1=="ULOGPFX" && mach_found==1 {
158       ulogpfx=1; printf "ulogpfx=%s\n", $2
159       next
160       }
161   section == 7 && NF == 1 {
162     if ( $1 == machine )
163       {mach_found = 1}
164     else { # reset mach_found for other machines
165       mach_found = 0
166       }
167     next
168     }
169   section == 7 && $1=="NLSADDR" && mach_found==1 {
170         printf "nlsaddr=%s\n", $2
171     next
172          }
173   section == 1 && $1 == "UID" {printf "uid=%s\n", $2 ;next }
174   section == 7 &&      $1=="BRIDGE" && mach_found==1 {
175                printf "bridge=%s\n", $2 }
176     END { # not defined ulogpfx
177      if ( ulogpfx == 0 ) {
178       printf "ulogpfx=%s/ULOG\n", appdir }
179          } ' machine=$machine appname=$appname
180         lang=`sed -e "s/=/ /g" -e "s/'//g" -e "s/;/ /" $ConfDir/$appname.tux | awk '
181           $1 == "LANG" {printf "lang=", $2}' `
182       }
183
184   get_tllog() {
185   tllogfname="$ConfDir/tlistenlog.$appname.$machine"
186   if [ -f $tllogfname ]
187    then
188     tllog=`cat $tllogfname|awk '$1 == "TLLOG" && $2 == machine { print $3 }' machine=$m
      achine`
189    else
190      tllog="$MADISON_TMP/tlistenlog.$appname.$machine"
191      echo "TLLOG $machine $tllog" > $tllogfname
192   fi
193   }
194
195
196   case $cmd in
197         appli)
198                ls -l $ConfDir 2> /dev/null | awk '
199                      sub(".tux$", "", $NF) {print $NF}'
200                ;;
201         isexist)
202                if [ -f $ConfDir/$1.tux ]
203                then
204                      echo "Yes"
205                else
206                      echo "No"
207                fi
208                ;;
209         setparam)
210                [ ! -d $ConfDir ] && mkdir -p $ConfDir
211                if [ -n "$2" ]
212                then
```

ANNEXE 2

```
213                     filename=$ConfDir/$2.tux
214                     while [ $# -gt 0 ]
215                     do
216                             echo "$1=\"$2\"; export $1"
217                             shift 2
218                     done > $filename
219             fi
220             ;;
221     discover)
222             [ -z "$1" ] && exit 1
223             filename=$ConfDir/$1.tux; shift
224             if [ -f $filename ]
225             then
226     #               sed -e 's/:/@@@/g' -e 's/#.*//' -e 's/ *; */"/g' $filename/ |
    awk '
227                     sed  -e 's/#.*//' -e 's/ *; */"/g' -e 's/:/#!:/g' $filename/
    | awk '
228                     BEGIN { field = "#promptW:promptP:promptPO:promptS:promptA:pr
    omptM:promptC:promptR:promptF"; value="::::::::" }
229                     /\=/ {
230                             for (i=1; i<= NF; i++) {
231                                 if(sub("=$", "", $i)) {
232                                    separator = ":"
233                                    field = field separator $i
234                                    value = value separator ${i+1}
235                                 }
236                             }
237                     }
238                     END {
239                             print field; print value
240                     }' FS='"'
241             else
242                     print '#\n'
243             fi
244             ;;
245     delappname)
246             if [ -n "$2" ]
247             then
248                     filename=$ConfDir/$2.tux
249                     if [ -f $filename ] && grep -q "$1=['\"]*$2" $filename
250                     then
251                             rm -f $filename ${filename}p
252                     else
253                             echo 'The file does not exist'
254                             echo '          or'
255                             echo 'The file is not an environment file'
256                             exit 1
257                     fi
258             fi
259             ;;
260     select)
261             if [ -n "$2" ]
262             then
263                     echo "$1='$2'; export $1" > "$Context"
264             fi
265             ;;
266     deselect)
267             rm -f "$Context"
268             ;;
269     selected)
270             APPNAME=""
271             [ -f $Context ] && . ./$Context
272             echo "$1$APPNAME"
273             ;;
274     isselected)
275             rm -f tuxconf.tmp.*
276             [ -f $Context ] && fgrep -q "APPNAME=" $Context && shift
277             echo $1
278             ;;
279     loadcf)
280             appname=$1
```

ANNEXE 2

```
281      boucle_status=0
282              cmd="\$TUXDIR/bin/tmloadcf -y $2 $3"
283              set_environ
284     echo "---- Loading Configuration Binary File ---"
285              remote_cmd
286      status=$?
287      if [ $status -ne 0 ]
288       then
289        exit $status
290       else
291  # maj fichier $Scanconf.tux machines
292         prog="$Env"'
293         $TUXDIR/bin/tmunloadcf
294         echo "\nexit $?"
295         '
296         #print -r "$prog" > prog
297         rsh "$MASTER" -l "$ADMIN" "$prog" > tuxconf.tmp.$appname
298         list_lmids=`cat tuxconf.tmp.$appname | sed -e "s/=/ /g" -e 's/"//g' -e "s/\*//
     "  | awk '
299           {line = $0}
300           $2 == "LMID" && machine == 1 {lmids = lmids $3 " "; next}
301           $1 == "GROUPS" && $2 == "" { machine=0; next}
302           $1 == "MACHINES" && $2 == "" { machine = 1; next}
303           END {if(sub("^exit ","", line)) {
304              print lmids
305              exit line}
306              exit -1 }'`
307         for machine in $list_lmids
308           do
309           echo "---- Updating $Scanconf on $machine ----\n"
310           get_tuxval > "appname.tux"
311           . ./appname.tux
312           log_prefix=`echo $ulogpfx | sed -e 's./. .g' | awk '
313               {print $NF} '`
314           log_dir=`echo $ulogpfx | sed -e 's./. .g' | awk '
315             {for (i=1; i< NF; i++) {
316               tempo = tempo "/" $i }}
317             END { print tempo}'`
318  #Build the 3 lines of $Scanconf for the application
319           prog="
320  [ -x $MADISON_BIN/security/updscantux ] &&
321  $MADISON_BIN/security/updscantux $appname $log_dir $log_prefix
322  echo \"\\nexit \$?\""
323             rsh "$uname" -l madison "$prog"  | awk '
324               NR == 1 {line = $0}
325               NR > 1  { print line; line = $0}
326               END {if(sub("^exit ","", line)) exit line; exit -1 }'
327         boucle_status=`expr $boucle_status + $? `
328        done
329       fi
330       exit $boucle_status
331             ;;
332        apppwd)
333             filename=$ConfDir/$1.tuxp
334             echo "Enter Application Password: \c"
335             OLDCONFIG=`stty -g`
336             stty -echo
337             read APP_PW
338             echo "\nRe-enter Application Password: \c"
339             read APP_PW_1
340             stty $OLDCONFIG
341             if [ "$APP_PW" != "$APP_PW_1" ]
342             then
343                   echo "\n\nPassword mismatch!"
344                   echo "Enter any character to exit and retry"
345                   read
346             else
347        #          PWencode "APP_PW=\"$APP_PW\"; export APP_PW" > $filename
348          #            APP_PW=`echo $APP_PW | sed -e "s/'/'\"'\"'/g"`
349        #          PWencode "APP_PW='$APP_PW'; export APP_PW" > $filename
350  tuxgetenv -s > $filename << !
```

ANNEXE 2

```
tuxgetenvp
$APP_PW
!
                fi
                ;;
        chksyntx)
                appname=$1
                cmd="\$TUXDIR/bin/tmloadcf -n $2"
                set_environ
                remote_cmd
                exit $?
                ;;
        displpc)
                appname=$1
                cmd="\$TUXDIR/bin/tmloadcf -c $2"
                set_environ
                remote_cmd
                exit $?
        ;;
        machine_network)
                appname=$1
                set_environ
                get_tuxconfig | \
                sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                        BEGIN { network=0 }
                        {line = $0}
                        NF == 1 { if (network == 1) print $1}
                        $1 == "NETWORK" { network = 1}
                        END {if(sub("^exit ","", line)) exit line; exit -1 }'
                exit $?
                ;;

        machine_machines)
                appname=$1
                set_environ
                get_tuxconfig | \
                sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                        BEGIN { machine=0 }
                        {line = $0}
                        $2 == "LMID" { if(machine == 1) print $3}
                        $1 == "GROUPS" { if( $2 == "" ) machine=0}
                        $1 == "MACHINES" { if( $2 == "" ) machine = 1}
                        END {if(sub("^exit ","", line)) exit line; exit -1 }'
                exit $?
                ;;
        group)
                appname=$1
                set_environ
                get_tuxconfig | \
                sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                        BEGIN { group=0 }
                        {line = $0}
                        $1 == "SERVERS" { group=0 }
                        $1 == "GROUPS"  { if($2 == "") group=1}
                        $2 == "LMID" && $4 == "GRPNO" { if(group) print $1}
                        END {if(sub("^exit ","", line)) exit line; exit -1 }'
                exit $?
                ;;
        svrname)
                appname=$1
                set_environ
                get_tuxconfig | \
                sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                        BEGIN { group=server=nb_of_distinct_svr_name=0 }
                        {line = $0}
                        $1 == "TMSNAME" { if ( group == 1) {
                            trouve = 0
                            if (nb_of_distinct_svr_name == 0) {
                              nb_of_distinct_svr_name=1
                              svr_names[nb_of_distinct_svr_name] = $2
                              print $2
```

ANNEXE 2

```
422                     } else {
423                       for (j=1; j<= nb_of_distinct_svr_name; j++) {
424                             if ( $2 == svr_names[j] ) {
425                                   trouve=1
426                             }
427                       }
428                       if (trouve == 0) {
429                             nb_of_distinct_svr_name += 1
430                             svr_names[nb_of_distinct_svr_name] = $2
431                             print $2
432                         }
433                       }
434                     }
435                 }
436                 $1 == "SERVERS" { if ($2 == "") {
437                       server=1
438                       group=0 }
439                 }
440                 $1 == "SERVICES" { if ($2== "") server=0}
441                 $1 == "GROUPS"   { if ($2 == "") group=1}
442                 $2 == "SRVGRP" {
443                     if((server == 1) && ( $4 == "SRVID")) {
444                       trouve = 0
445                       if (nb_of_distinct_svr_name == 0) {
446                           nb_of_distinct_svr_name = 1
447                           svr_names[nb_of_distinct_svr_name] = $1
448                           print $1
449                       } else {
450                           for(j=1; j<= nb_of_distinct_svr_name; j++) {
451                             if ( $1 == svr_names[j] ) {
452                                   trouve=1
453                             }
454                           }
455                           if(trouve == 0) {
456                             nb_of_distinct_svr_name += 1
457                             svr_names[nb_of_distinct_svr_name] = $1
458                             print $1
459                           }
460                       }
461                     }
462                 }
463                 END {if(sub("^exit ","", line)) exit line; exit -1 }'
464         exit $?
465         ;;
466     svrseq)
467         appname=$1
468         set_environ
469         get_tuxconfig | \
470         sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
471                 BEGIN { server=0; nb_of_distinct_svr_seq=0 }
472                 {line = $0}
473                 $1 == "SEQUENCE" && server == 1 {
474                     trouve = 0
475                     if (nb_of_distinct_svr_seq == 0) {
476                       nb_of_distinct_svr_seq=1
477                       svr_seqs[nb_of_distinct_svr_seq] = $2
478                       print $2
479                     } else {
480                       for (j=1; j<= nb_of_distinct_svr_seq; j++) {
481                             if ( $2 == svr_seqs[j] ) {
482                                   trouve=1
483                             }
484                       }
485                       if (trouve == 0) {
486                             nb_of_distinct_svr_seq += 1
487                             svr_seqs[nb_of_distinct_svr_seq] = $2
488                             print $2
489                       }
490                     }
491                 }
492                 $1 == "SERVERS" {  if($2 == "") server=1}
```

ANNEXE 2

```
493                 $1 == "SERVICES" { if($2 == "") server=0}
494                 END {if(sub("^exit ","", line)) exit line; exit -1 }'
495         exit $?
496         ;;
497     svrId)
498         appname=$1
499         set_environ
500         get_tuxconfig | \
501         sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
502                 BEGIN { server=0; nb_of_distinct_svr_Id=0 }
503                 {line = $0}
504                 $2 == "SRVGRP" && $4 == "SRVID" &&  server == 1 {
505                     trouve = 0
506                     if (nb_of_distinct_svr_Id == 0) {
507                        nb_of_distinct_svr_Id=1
508                        svr_Ids[nb_of_distinct_svr_Id] = $5
509                        print $5
510                     } else {
511                        for (j=1; j<= nb_of_distinct_svr_Id; j++) {
512                              if ( $5 == svr_Ids[j] ) {
513                                    trouve=1
514                              }
515                        }
516                        if (trouve == 0) {
517                              nb_of_distinct_svr_Id += 1
518                              svr_Ids[nb_of_distinct_svr_Id] = $5
519                              print $5
520                        }
521                     }
522                 }
523                 $1 == "SERVERS" {  if($2 == "") server=1}
524                 $1 == "SERVICES" { if($2 == "") server=0}
525                 END {if(sub("^exit ","", line)) exit line; exit -1 }'
526         exit $?
527         ;;
528     discover_conf)
529         machine=$2
530         appname=$1
531         set_environ
532         get_tuxconfig | \
533         sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\\/0/' -e "s/\*//"  | awk '
534                 BEGIN {field = "#" }
535                 {line = $0}
536                 $1 == "UID" {
537                       field = field separator $1
538                       value = value separator $2
539                       separator = ":"
540                 }
541                 $1 == "GID" {
542                       field = field separator $1
543                       value = value separator $2
544                       separator = ":"
545                 }
546
547                 $1 == "BRIDGE" && network == 1 && mach_found == 1 {
548                       field = field separator $1
549                       value = value separator $2
550                 }
551                      $1 == "NLSADDR" && network == 1 && mach_found == 1 {
552                             field = field separator $1
553                             value = value separator $2
554                        network = 0
555                        mach_found = 0
556                      }
557                      $1 == "TLLOG" && $2 == machine {
558                             field = field separator $1
559                        value = value separator $3
560                      }
561
562                 $1 == machine {mach_found = 1}
563                 $1 == "NETWORK"  { network = 1}
```

ANNEXE 2

```
564                     END {
565                           print field; print value
566                           if(sub("^exit ","", line)) exit line; exit -1
567                     }' "machine=$machine"
568             exit $?
569             ;;
570         chglisten)
571                appname=$1
572                machine=$2
573                shift 2
574                if [ $# -gt 0 ]
575                 then
576                   echo "TLLOG $machine $1" > $ConfDir/tlistenlog.$appname.$machine
577                fi
578                exit $?
579             ;;
580         chklistscript)
581                appname=$1
582                machine=$2
583                set_environ
584                get_tuxval > "appname.tux"
585                get_tllog
586                . ./appname.tux
587                prog="
588                if [ -f $appdir/tlisten.$appname.$machine ]
589                 then
590                   cat $appdir/tlisten.$appname.$machine
591                   echo \"\\nexit 0\"
592                 else
593                   echo \"\\nexit 1\"
594                fi"
595                if [ -z "$uname" ]
596                  then
597                    print "Host $machine not found"
598                    exit 1
599                fi
600                rm -f tlscript.$appname.$machine
601                rsh "$uname" -l "$ADMIN" "$prog" | tee tlscript.$appname.$machine > /
    dev/null
602                [ $? -ne 0 ] && exit 1
603                [ -s tlscript.$appname.$machine ] && cat tlscript.$appname.$machine |
    awk '
604                  END { if ( $2 == "1" ) exit -1} '
605                [ $? -eq -1 ] && exit 1
606                [ -s tlscript.$appname.$machine ] && cat tlscript.$appname.$machine |
    \
607                awk '
608                 $1 ~ "tlisten" {
609                     mismatch = 0
610                     fexec=sprintf("%s/bin/tlisten", tuxdir)
611                     if ($1 != fexec) {
612                        print "tlisten command full pathnames mismatch"
613                        printf "\tscript:\t%s\n", $1
614                        printf "\tconfig:\t%s\n", fexec
615                        mismatch +=1
616                        }
617                     for (i=2; i <= NF; i++) {
618                        if (( $i == "-d") && ($(i+1) != bridge)) {
619                           print "BRIDGE values mismatch"
620                           printf "\tscript:\t%s\n",$(i+1)
621                           printf "\tconfig:\t%s\n",bridge
622                           mismatch +=1
623                           }
624                         if (( $i == "-l") && ($(i+1) !=nlsaddr)) {
625                           print "NLSADDR values mismatch"
626                           printf "\tscript:\t%s\n",$(i+1)
627                           printf "\tconfig:\t%s\n", nlsaddr
628                           mismatch +=1
629                           }
630                         if (( $i == "-u") && ($(i+1) != uid)) {
631                           print "UID values mismatch"
```

ANNEXE 2

```
632                          printf "\tscript:\t%s\n", $(i+1)
633                          printf "\tconfig:\t%s\n",uid
634                          mismatch +=1
635                          }
636                        if (( $i == "-L") && ($(i+1) !=tllog)) {
637                          print "LOGFILE values mismatch"
638                          printf "\tscript:\t%s\n", $(i+1)
639                          printf "\tconfig:\t%s\n", tllog
640                          mismatch +=1
641                          }
642                        }}
643                  END {
644                     if ( mismatch == 0 )
645                      printf "Script File is up-to-date for %s\n",machine
646                     else
647                      printf "\nScript File is NOT up-to-date for %s\n",machine
648                     } ' tllog=$tllog machine=$machine bridge=$bridge \
649                          nlsaddr=$nlsaddr uid=$uid tuxdir=$tuxdir
650               exit $?
651               ;;
652       startlistproc)
653             appname=$1; shift
654             list="$*"
655             set_environ
656             boucle_status=0
657             exit_status=0
658             for machine in $list
659             do
660                echo "\n------- Machine: $machine -------\n"
661                get_tuxval > "appname.tux"
662                   get_tllog
663                   . ./appname.tux
664                   prog1="
665                   TUXDIR=$tuxdir; export TUXDIR
666                ROOTDIR=$tuxdir; export ROOTDIR  # V4
667                   APPDIR=$appdir; export APPDIR
668                   TUXCONFIG=$tuxconfig; export TUXCONFIG
669                   PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
670                   LANG=$lang; export LANG
671                   LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
672                   COLUMNS=200; export COLUMNS
673                   ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {
    exit 1}'
674                   if [ \$? = 1 ]
675                     then
676                        echo \"Listener already running on $machine\"
677                        echo exit 0
678                      exit 0
79                          fi
680                   if [ -f $appdir/tlisten.$appname.$machine ]
681                   then
682                        . $appdir/tlisten.$appname.$machine
683                      ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nls
    addr\" {exit 1}'
684                      if [ \$? = 1 ]
685                      then
686                         echo \"Listener started on $machine\"
687                         echo exit 0
688                      else
689                        echo \"Listener starting failed on $machine !!!\"
690                         echo exit 1
691                      fi
692                    else # create the script file & exec it
693                         echo \"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L
    $tllog\" > $appdir/tlisten.$appname.$machine
694                      chmod ug+x $appdir/tlisten.$appname.$machine
695                      . $appdir/tlisten.$appname.$machine
696                      ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsadd
    r\" {exit 1}'
697                      if [ \$? = 1 ]
698                       then
```

ANNEXE 2

```
                          echo \"Listener started on $machine\"
                          echo exit 0
                        else
                            echo \"Listener starting failed on $machine !!!\"
                            echo exit 1
                        fi
                    fi"
        #echo "$prog1" > prog1
        if [ -z "$uname" ]
           then
              print "Host $machine not found"
              exit 1
        fi
      rsh "$uname" -l "$ADMIN" "$prog1" | awk '
            NR == 1 {line = $0}
            NR > 1 { print line; line = $0 }
            END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
      boucle_status=`expr $boucle_status \| $? `
          done
           exit $boucle_status
    ;;
  stoplistproc)
      appname=$1; shift
      list="$*"
      set_environ
      boucle_status=0
      exit_status=0
      for machine in $list
         do
          echo "\n------- Machine: $machine -------\n"
            get_tuxval > "appname.tux"
            . ./appname.tux
            prog1="
          COLUMNS=200; export COLUMNS
          ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {print \$
2; exit 0 }' | read pid
          if [ -n \"\$pid\" ]
            then
              kill -9 \$pid > /dev/null
              status=\$?
              if [ \$status -eq 0 ]
              then
                    echo \"Process \$pid killed on $machine\"
                    echo exit 0
              else
                    echo \"Failed to stop listener on $machine!!!\"
                    echo exit 1
              fi
          else
              echo \"No Listener running on $machine\"
              echo exit 1
          fi"
            if [ -z "$uname" ]
               then
                  print "Host $machine not found"
                  exit 1
            fi
          rsh "$uname" -l "$ADMIN" "$prog1" | awk '
              NR == 1 {line = $0}
              NR > 1 { print line; line = $0 }
              END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
          boucle_status=`expr $boucle_status \| $? `
            done
       exit $boucle_status
    ;;

       runninglist)
        appname=$1
        boucle_status=0
        set_environ
        list_lmids=`get_tuxconfig | \
```

ANNEXE 2

```
769           sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\\/0/' -e "s/\*//"  | awk '
770              BEGIN { network=0 }
771              {line = $0}
772              NF == 1 { if (network == 1) print $1}
773              $1 == "NETWORK" { network = 1}
774              END {if(sub("^exit ","", line)) exit line; exit -1 }' `
775           for machine in $list_lmids
776             do
777               get_tuxval > "appname.tux"
778               . ./appname.tux
779               prog1="
780               TUXDIR=$tuxdir; export TUXDIR
781               LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
782               ROOTDIR=$tuxdir; export ROOTDIR  # V4
783               APPDIR=$appdir; export APPDIR
784               TUXCONFIG=$tuxconfig; export TUXCONFIG
785               PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
786               LANG=$lang; export LANG
787               COLUMNS=200; export COLUMNS
788               ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {print
    \$2}' | read pid
789               if [ -n \"\$pid\" ]
790                then
791                  echo \"Listener running on $machine: pid = \$pid\"
792                  echo exit 0
793                else
794                  echo \"No Listener running on $machine\"
795                  echo exit 0
796               fi"
797               if [ -z "$uname" ]
798                 then
799                   print "Host $machine not found"
800                   exit 1
801               fi
802               rsh "$uname" -l "$ADMIN" "$prog1" | awk '
803                NR == 1 {line = $0}
804                NR > 1  { print line; line = $0}
805                END { if (sub("^exit ", "", line)) exit line; print line; exit -1} '
806                boucle_status=`expr $boucle_status \| $?`
807             done
808             exit $boucle_status
809           ;;
810           updtlistscript)
811           appname=$1
812           machine=$2
813           set_environ
814           get_tllog
815           get_tuxval > "appname.tux"
816           . ./appname.tux
817           prog="
818           echo \"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L $tllog\" >  $app
    dir/tlisten.$appname.$machine
819           chmod ug+x $appdir/tlisten.$appname.$machine
820           echo exit \$?"
821           if [ -z "$uname" ]
822             then
823               print "Host $machine not found"
824               exit 1
825           fi
826           rsh "$uname" -l "$ADMIN" "$prog" | awk '
827             NR == 1 {line = $0}
828             NR > 1 { print line; line = $0 }
829             END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
830           exit $?
831           ;;
832         tuxBootEnt)
833               appname=$1; shift
834               cmd="\$TUXDIR/bin/tmboot -y $@"
835               set_environ
836               remote_cmd
837               exit $?
```

ANNEXE 2

```
838             ;;
839         tuxShutEnt)
840             appname=$1; shift
841             cmd="\$TUXDIR/bin/tmshutdown -y"
842             set_environ
843             remote_cmd
844             exit $?
845             ;;
846         tuxBootAllMach)
847             appname=$1; shift
848             cmd="\$TUXDIR/bin/tmboot -y -A  $@"
849             set_environ
850             remote_cmd
851             exit $?
852             ;;
853         tuxShutAllMach)
854             appname=$1; shift
855             cmd="\$TUXDIR/bin/tmshutdown -y -A $@"
856             set_environ
857             remote_cmd
858             exit $?
859             ;;
860         tuxShut)
861             appname=$1; shift
862             cmd="\$TUXDIR/bin/tmshutdown -y $@"
863             set_environ
864             remote_cmd
865             exit $?
866             ;;
867         tuxShutAdmMast)
868             appname=$1; shift
869             cmd="\$TUXDIR/bin/tmshutdown -y -M $@"
870             set_environ
870             remote_cmd
872             exit $?
873             ;;
874         tuxShutSvrSect)
875             appname=$1; shift
876             cmd="\$TUXDIR/bin/tmshutdown -y -S $@"
877             set_environ
878             remote_cmd
879             exit $?
880             ;;
881         tuxBootAdmMast)
882             appname=$1; shift
883             cmd="\$TUXDIR/bin/tmboot -y -M $@"
884             set_environ
885             remote_cmd
886             exit $?
887             ;;
888         tuxBoot)
889             appname=$1; shift
890             cmd="\$TUXDIR/bin/tmboot -y $@"
891             set_environ
892             remote_cmd
893             exit $?
894             ;;
895         tuxShutdown)
896             appname=$2
897             cmd="\$TUXDIR/bin/tmshutdown -y $1"
898             set_environ
899             remote_cmd
900             exit $?
901             ;;
902         tuxBootSvrSct)
903             appname=$1; shift
904             cmd="\$TUXDIR/bin/tmboot -y -S $@"
905             set_environ
906             remote_cmd
907             exit $?
908             ;;
```

ANNEXE 2

```
909         tuxBootBBL)
910                 #echo $*
911               appname=$1; shift
912               cmd="\$TUXDIR/bin/tmboot -y $@"
913               set_environ
914               remote_cmd
915               exit $?
916               ;;
917         tuxShowBooted)
918               appname=$1; shift
919               cmd="(echo psr; echo quit)|\$TUXDIR/bin/tmadmin"
920               set_environ
921               remote_cmd
922               exit $?
923               ;;
924         tuxminIPC)
925               appname=$1; shift
926               cmd="\$TUXDIR/bin/tmboot -y -c $@"
927               set_environ
928               remote_cmd
929               exit $?
930               ;;
931         tuxShutPart)
932                  exit_status=0
933               appname=$1;
934               machine=$2; shift
935               set_environ
936               get_tuxconfig | \
937               sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
938                     $1 == "APPDIR" && mach_section == 1 && mach_found == 1 {
939                           print "APPDIR " $2 > "appname.tux"
940                           mach_section = 0
941                           mach_found = 0
942                     }
943                     $1 == "TUXCONFIG" && mach_section==1 && mach_found==1 {
944                           print "TUXCONFIG " $2 > "appname.tux"
945                     }
946                     $1 == "MACHINES" {mach_section = 1}
947                     $2 == "LMID" && mach_section == 1 && $3 == machine {
948                           print "MACHINE " $1 > "appname.tux"
949                           mach_found = 1
950                     }
951                     $1 == "TUXDIR" && mach_section==1 && mach_found==1 {
952                           print "TUXDIR " $2 > "appname.tux"
953                     }
954               ' "machine=$machine" "appname=$appname"
955               if [ $? != 0 ]
956               then
957                     exit 1
958               fi
959             appdir=`awk '$1 == "APPDIR" {print $2}' appname.tux`
960             tuxconfig=`awk '$1 == "TUXCONFIG" {print $2}' appname.tux`
961             uname=`awk '$1 == "MACHINE" {print $2}' appname.tux`
962             rootdir=`awk '$1 == "TUXDIR" {print $2}' appname.tux`
963             lang=`sed -e 's/=/ /g' -e 's/;/ /g' $ConfDir/$appname.tux |
964                   awk '$1 == "LANG" {print $2}'`
965             prog1="TUXDIR=$rootdir; export TUXDIR
966                    APPDIR=$appdir; export APPDIR
967                    LIBPATH=${LIBPATH}:$rootdir/lib; export LIBPATH
968                    TUXCONFIG=$tuxconfig; export TUXCONFIG
969                    LANG=$lang; export LANG
970                    PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
971                    $TUXDIR/bin/tmshutdown -y -P $@
972                    echo \$? > /tmp/rem$appname.$machine.tux"
973             if [ -z "$uname" ]
974               then
975                  print "Host $machine not found"
976                  exit 1
977             fi
978             rsh $uname -l "$ADMIN" "$prog1"
979             rsh_status=`echo $?`
```

ANNEXE 2

```
980               if [ "$rsh_status" -eq "0" ]
981                 then
982                  status=`rsh $uname -l "$ADMIN" "cat /tmp/rem$appname.$machine.tux"`
983                  rsh $MASTER -l "$ADMIN" "rm /tmp/rem$appname.$machine.tux" 2> /dev/nul
     l
984                  rsh $uname -l "$ADMIN" "rm /tmp/rem$appname.$machine.tux"  2> /dev/nul
     l
985               fi
986               if [ "$status" -ne "0" ]
987                 then
988                    exit_status=`expr $exit_status + 1`
989               fi
990               if [ "$exit_status" -ne "0" -o "$rsh_status" -ne "0" ]
991                 then
992                   exit 1
993               fi
994       ;;
995       loadfshm)
996       appname=$1; machine=$2; shift 2
997       set_environ
998       get_tuxval > "appname.tux"
999       . ./appname.tux
1000      prog="
1001      TUXDIR=$tuxdir; export TUXDIR
1002      ROOTDIR=$tuxdir; export ROOTDIR
1003      LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
1004      LANG=$lang; export LANG
1005      $tuxdir/bin/loadfiles $@
1006      echo \"\nexit \$?\""
1007      if [ -z "$uname" ]
1008        then
1009          print "Host $machine not found"
1010          exit 1
1011      fi
1012      rsh "$uname" -l "$ADMIN" "$prog" | awk '
1013         NR == 1 {line = $0}
1014         NR > 1 { print line; line = $0 }
1015         END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
1016      ;;
1017      Unloadcf)
1018      appname=$1
1019      set_environ
1020      cmd="\$TUXDIR/bin/tmunloadcf"
1021      if [ $# -eq 2 ]
1022        then
1023         filename=$2
1024         remote_cmd > "$filename"
1025      else
1026         remote_cmd
1027      fi
1028      exit $?
1029      ;;
1030     *)
1031      echo "Command $1 does not exist"
1032      exit 1
1033      ;;
1034  esac
```

REVENDICATIONS

1. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions, basée sur un fichier binaire de configuration (TUXCONFIG) caractérisé en ce que ledit procédé comporte:

- une étape de récupération d'informations relatives à ladite application dans un fichier de configuration d'une machine maître (Mm),

- une étape de vérification de la consistance de ladite application mise en oeuvre sur une machine donnée.

2. Procédé selon la revendication 1, caractérisé en ce qu'il comprend une étape de gestion d'au moins un module d'écoute (3) d'une machine quelconque de l'application à partir d'une autre machine.

3. Procédé selon la revendication 1, caractérisé en ce que les informations concernant ladite application distribuée sont directement prélevées dans le fichier de configuration actif de la machine maître.

4. Procédé selon la revendication 1, caractérisé en ce que l'étape de vérification de consistance de ladite application consiste en une comparaison entre des informations issues du fichier de configuration de la machine maître et des informations issues de ladite application courante mise en oeuvre sur une machine donnée.

5. Procédé selon la revendication 2, caractérisé en ce que ladite gestion des modules d'écoute consiste à lancer et à arrêter au moins un module d'écoute, à afficher des informations concernant au moins un module d'écoute, à modifier le journal d'au moins un module d'écoute, à vérifier le script d'au moins un module d'écoute et/ou à mettre à jour le script d'au moins un module d'écoute.

6. Procédé selon la revendication 2, caractérisé en ce qu'il comprend une étape de lancement et d'arrêt d'un module d'écoute mis en oeuvre sur une première machine, cette étape étant mise en œuvre par un

administrateur utilisant une deuxième machine distincte de la première, appartenant au même réseau que la première machine.

7. Procédé selon la revendication 2, caractérisé en ce qu'il comprend une étape d'activation simultanée de plusieurs modules d'écoute.

8. Procédé selon la revendication 1, caractérisé en ce qu'il comporte une étape de décompilation du fichier de configuration actif de la machine maître.

9. Procédé selon la revendication 2, caractérisé en ce que les étapes du procédé sont mises en œuvre par l'intermédiaire d'une interface graphique comprenant au moins une icône, au moins un menu, et au moins une boîte de dialogue.

10. Procédé selon la revendication 9, caractérisé en ce que les menus de l'interface graphique sont structurés sous forme d'arborescence et l'actionnement d'un menu provoque l'affichage d'une liste de valeurs de la configuration courante, sélectionnable par l'utilisateur.

11. Procédé selon la revendication 4, caractérisé en ce que lorsque le fichier contenant des informations sur ladite application mise en oeuvre sur une machine donnée (tlog) est inexistant le procédé le génère automatiquement pour pouvoir l'utiliser lors du prochain lancement des modules d'écoute (3).

12. Procédé selon la revendication 6, caractérisé en ce lesdites informations affichées concernant au moins un module d'écoute(3) comprennent au moins le nom de ladite application, le nom logique de la machine (LMID)sur laquelle ladite application est exécutée, l'identification de l'utilisateur (UID) de ladite application, l'adresse utilisée par le module d'écoute (NLSADRR), le chemin d'accès au réseau de ladite application, le chemin d'accès au fichier journal dudit module d'écoute (LLFPN).

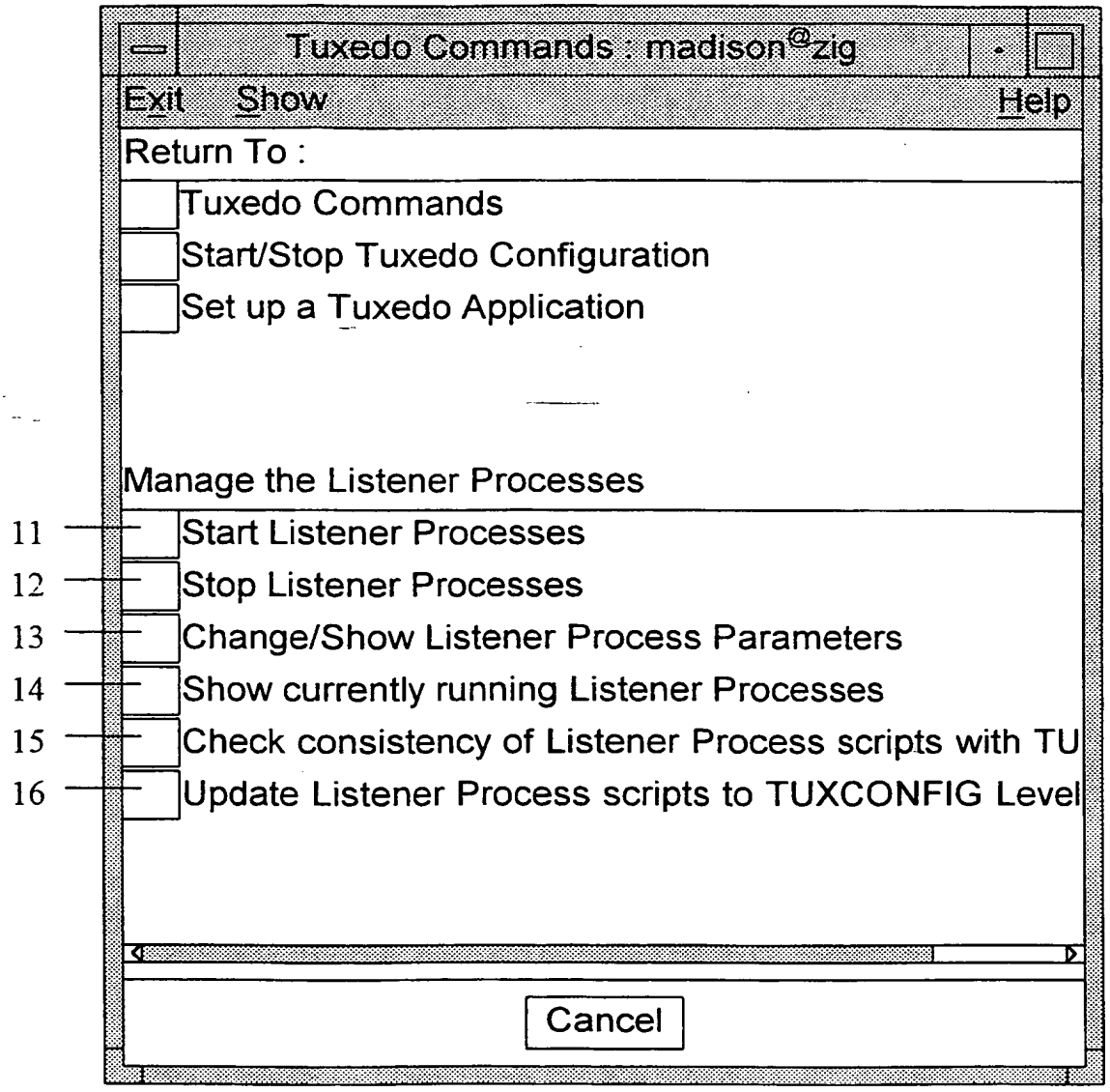

Tuxedo Commands : madison@zig
Exit Show Help
Return To :
Tuxedo Commands
Start/Stop Tuxedo Configuration
Set up a Tuxedo Application
Manage the Listener Processes
11
Start Listener Processes
12
Stop Listener Processes
13
Change/Show Listener Process Parameters
14
Show currently running Listener Processes
15
Check consistency of Listener Process scripts with TU
16
Update Listener Process scripts to TUXCONFIG Level
Cancel

FIG. 1

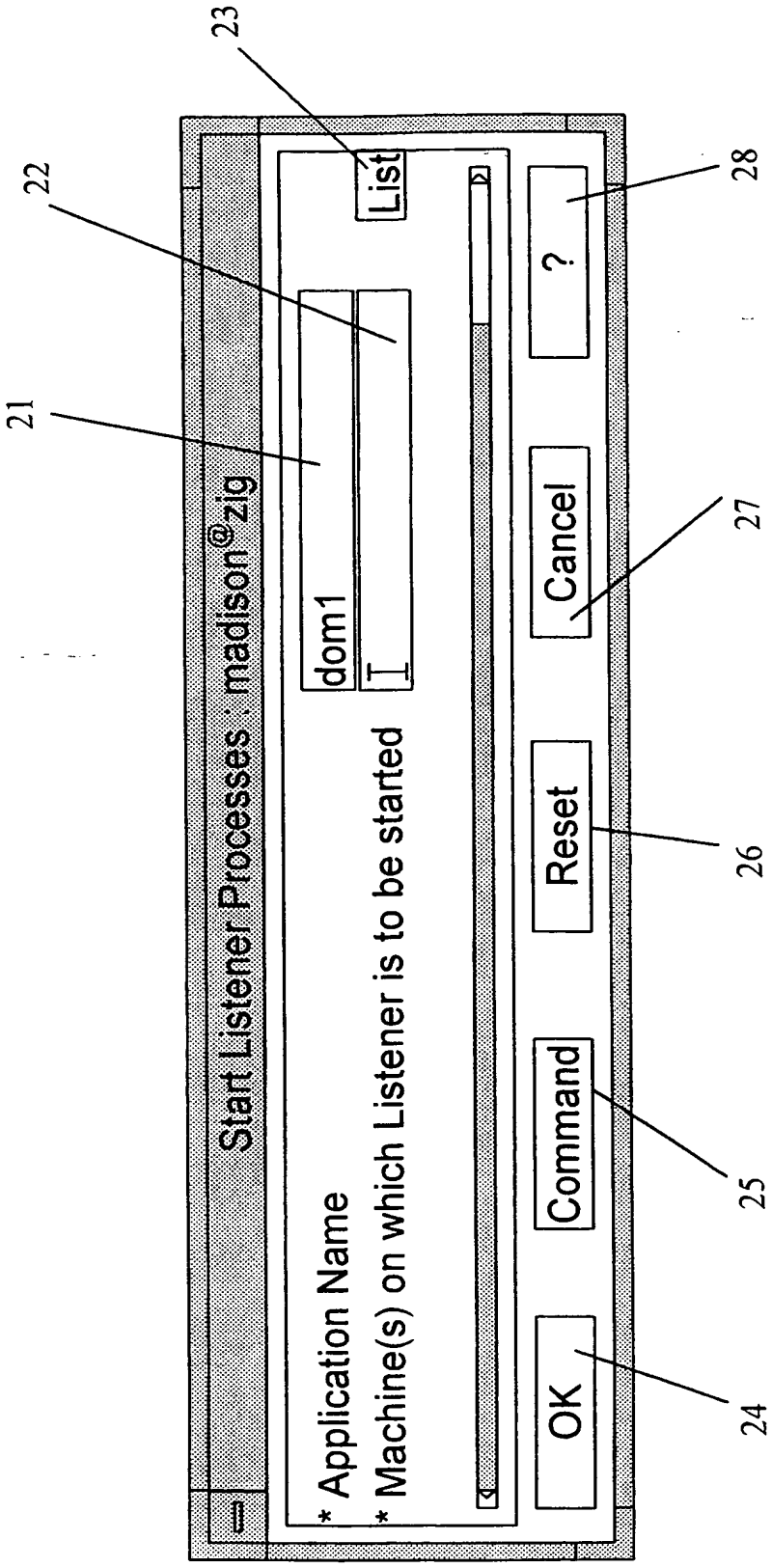

Start Listener Processes : madison@zig
* Application Name
* Machine(s) on which Listener is to be started
dom1
List
?
Cancel
Reset
Command
OK
21
22
23
24
25
26
27
28

FIG. 2

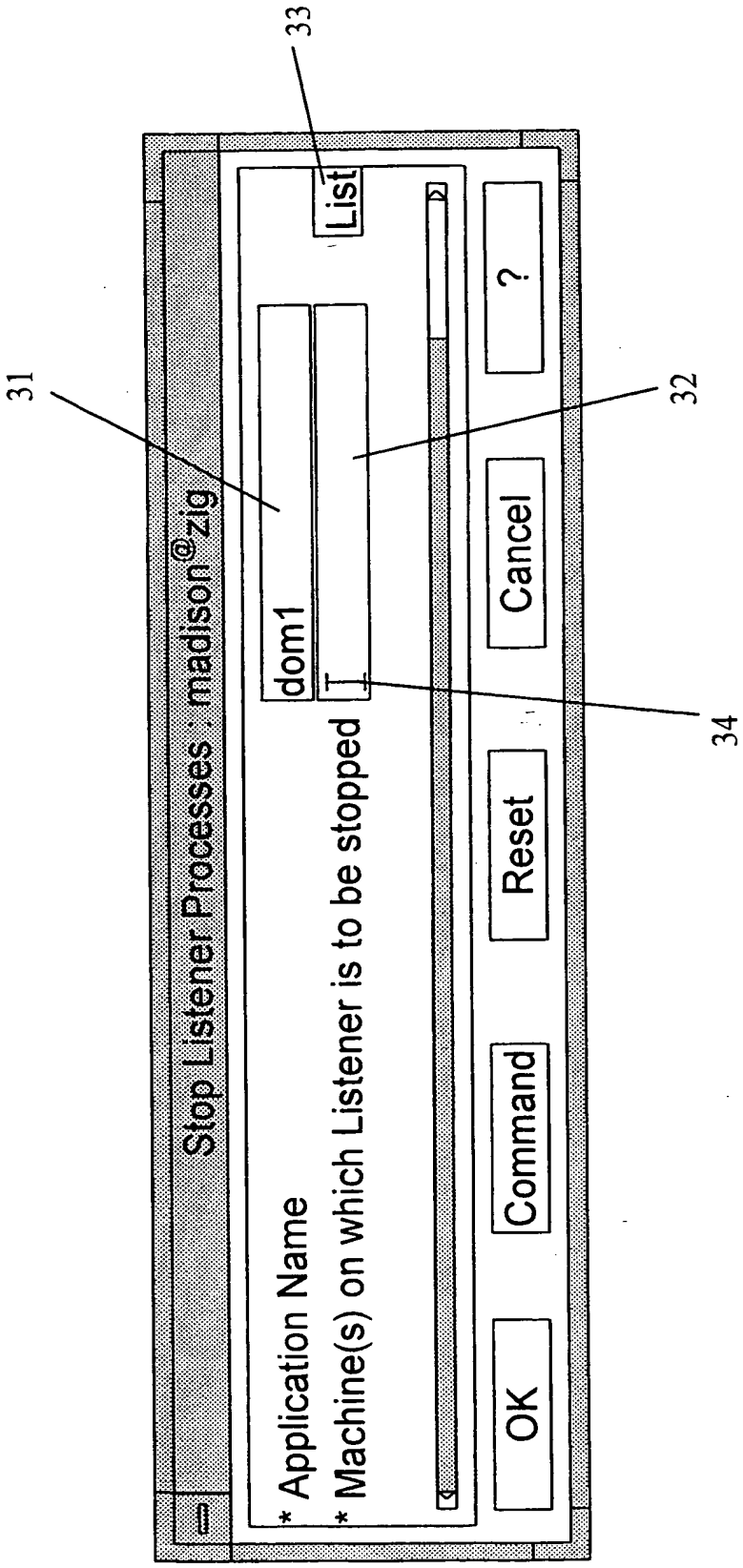

Stop Listener Processes : madison@zig
* Application Name
* Machine(s) on which Listener is to be stopped
dom1
List
?
Cancel
Reset
Command
OK
31
32
33
34

FIG. 3

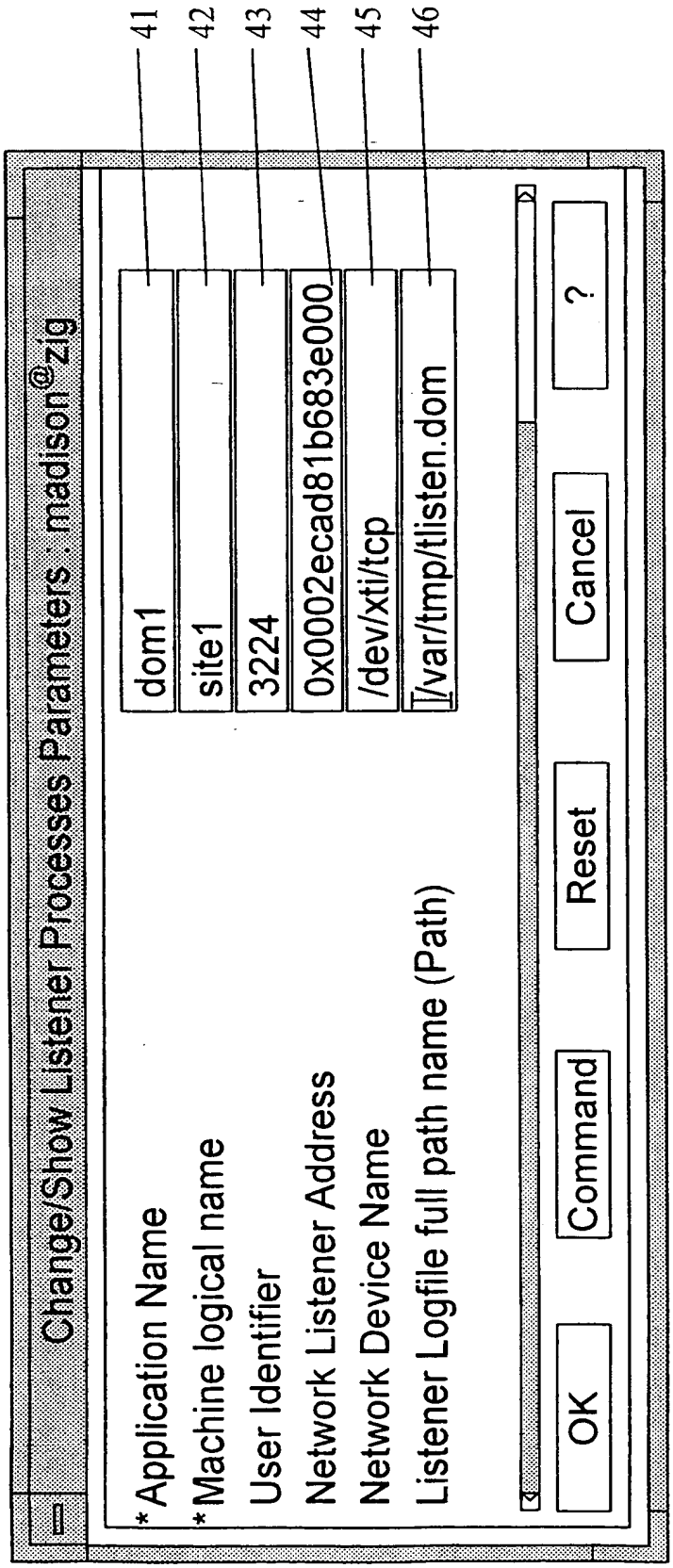

Change/Show Listener Processes Parameters : madison@zig
* Application Name
* Machine logical name
User Identifier
Network Listener Address
Network Device Name
Listener Logfile full path name (Path)
dom1
site1
3224
0x0002ecad81b683e000
/dev/xti/tcp
/var/tmp/tlisten.dom
OK
Command
Reset
Cancel
?
41
42
43
44
45
46

FIG. 4

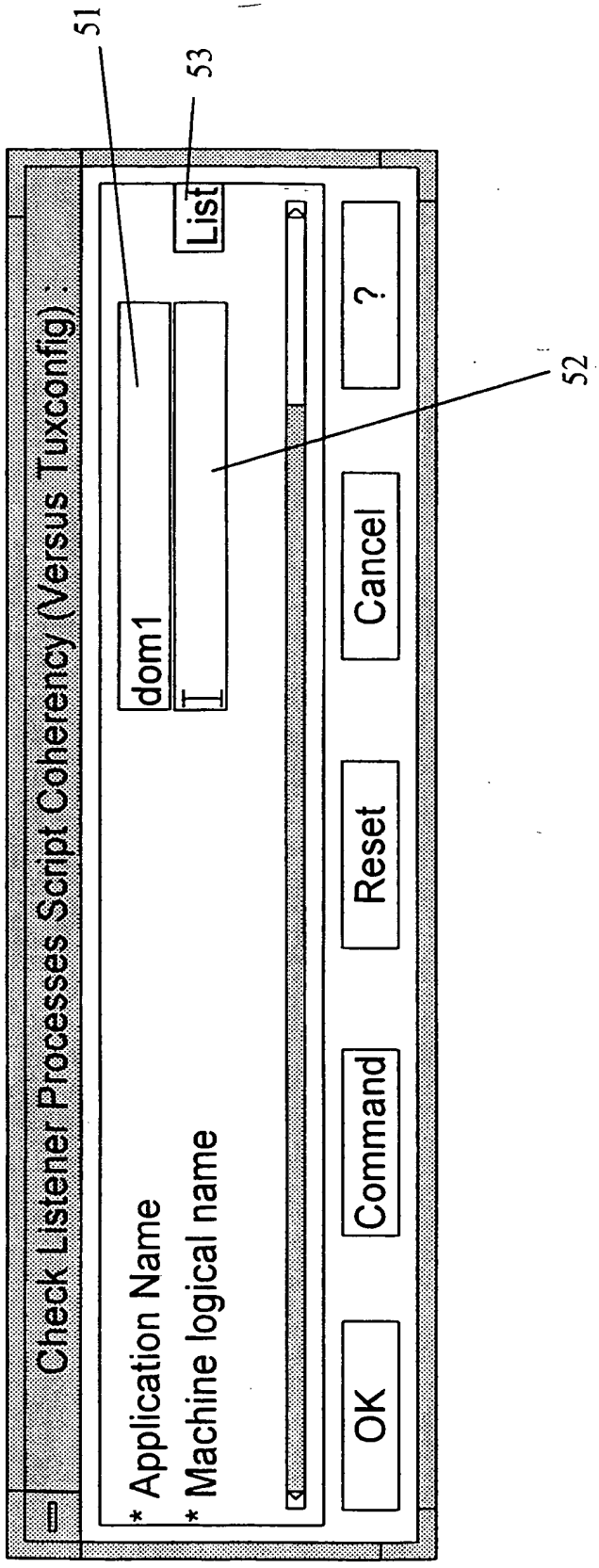

51
53
52
Check Listener Processes Script Coherency (Versus Tuxconfig)
* Application Name
* Machine logical name
dom1
List
OK
Command
Reset
Cancel
?

FIG. 5

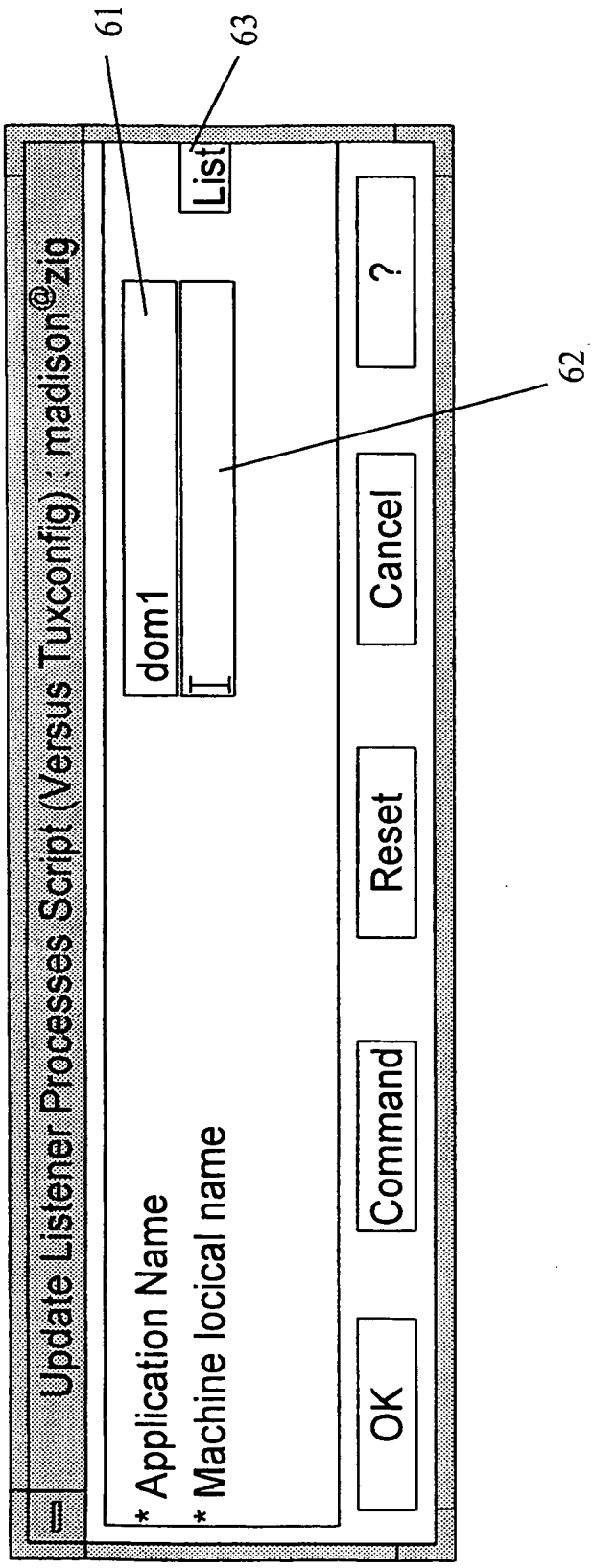

61
63
List
Update Listener Processes Script (Versus Tuxconfig) : madison@zig
dom1
62
* Application Name
* Machine locical name
OK
Command
Reset
Cancel
?

FIG. 6

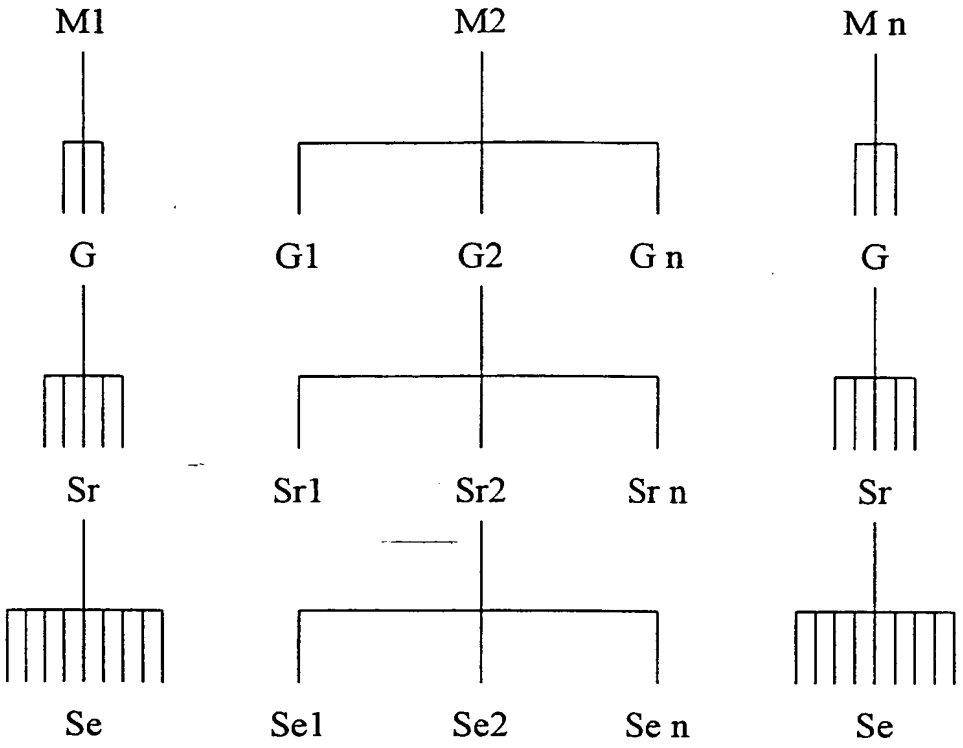

M1
M2
M n
G
G1
G2
G n
G
Sr
Sr1
Sr2
Sr n
Sr
Se
Se1
Se2
Se n
Se

FIG. 7

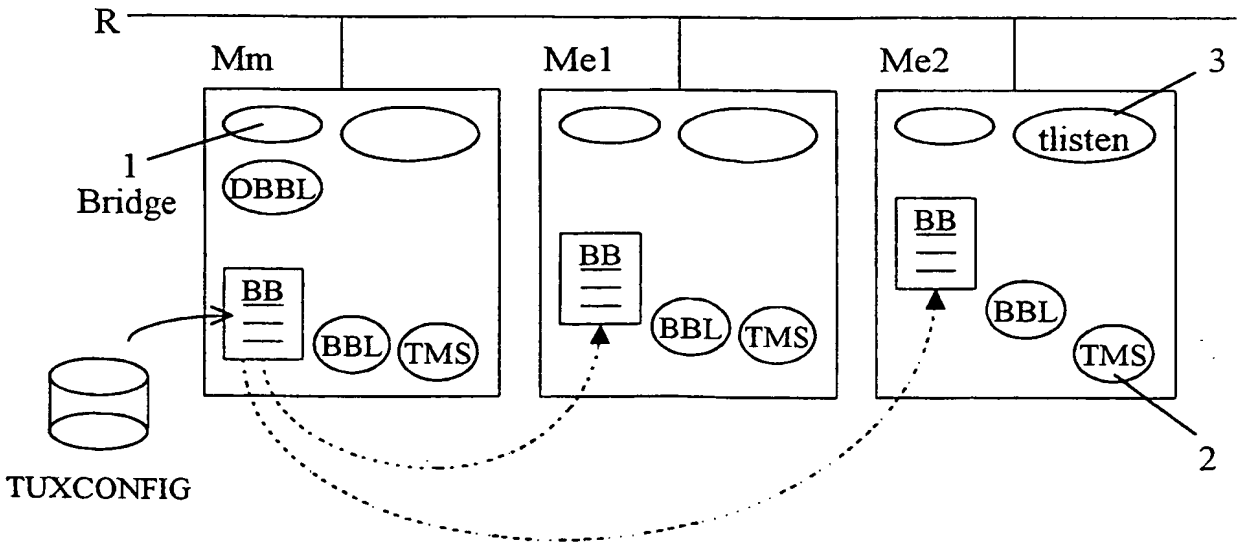

R
Mm
Me1
Me2
3
1
Bridge
DBBL
tlisten
BB
BBL
TMS
BB
BBL
TMS
BB
BBL
TMS
2
TUXCONFIG

FIG. 8

INTERNATIONAL SEARCH REPORT

I. national Application No
PCT/FR 98/02886

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/46

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	BERNSTEIN P A: "TRANSACTION PROCESSING MONITORS" COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY, vol. 33, no. 11, 1 November 1990, pages 75-86, XP000110643	1-4,6-8
A	see page 85, left-hand column, line 15 - middle column, line 42; figures 1,2	5,12
Y	US 5 557 735 A (PINKSTON II WILLIAM J ET AL) 17 September 1996 see claims 1-8; figure 2	1-4,6-8

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 June 1999	17/06/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kingma, Y

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

national Application No
PCT/FR 98/02886

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	GUPTA P: "HP ENCINA/7000: MIDDLEWARE FOR CONSTRUCTING TRANSACTION PROCESSING APPLICATIONS" HEWLETT-PACKARD JOURNAL, vol. 46, no. 6, 1 December 1995, pages 61-74, XP000581127 see page 72, right-hand column, line 12 - line 30	1,5,12

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

national Application No
PCT/FR 98/02886

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5557735 A	17-09-1996	NONE	

RAPPORT DE RECHERCHE INTERNATIONALE

Demande Internationale No
PCT/FR 98/02886

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 6 G06F9/46

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
CIB 6 G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si réalisable, termes de recherche utilisés)

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie °	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
Y	BERNSTEIN P A: "TRANSACTION PROCESSING MONITORS" COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY, vol. 33, no. 11, 1 novembre 1990, pages 75-86, XP000110643	1-4,6-8
A	voir page 85, colonne de gauche, ligne 15 - colonne du milieu, ligne 42; figures 1,2	5,12
Y	US 5 557 735 A (PINKSTON II WILLIAM J ET AL) 17 septembre 1996 voir revendications 1-8; figure 2	1-4,6-8

-/--

[X] Voir la suite du cadre C pour la fin de la liste des documents

[X] Les documents de familles de brevets sont indiqués en annexe

° Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent
"E" document antérieur, mais publié à la date de dépôt international ou après cette date
"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)
"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens
"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée
"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention
"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément
"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier
"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
11 juin 1999	17/06/1999

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kingma, Y

Formulaire PCT/ISA/210 (deuxième feuille) (juillet 1992)

RAPPORT DE RECHERCHE INTERNATIONALE

L ande Internationale No
PCT/FR 98/02886

C.(suite) DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie	Identification des documents cités, avec le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	GUPTA P: "HP ENCINA/7000: MIDDLEWARE FOR CONSTRUCTING TRANSACTION PROCESSING APPLICATIONS" HEWLETT-PACKARD JOURNAL, vol. 46, no. 6, 1 décembre 1995, pages 61-74, XP000581127 voir page 72, colonne de droite, ligne 12 - ligne 30	1,5,12

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

ıande Internationale No
PCT/FR 98/02886

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 5557735 A	17-09-1996	AUCUN	

FIGURE POUR L'ABREGE